Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2017

Maria João
Pinho Brandão

Modelação Comportamental da Camada Física NB-IoT - Uplink

Behavioral Modeling of the NB-IoT Uplink Physical Layer

Maria João
Pinho Brandão

Modelação Comportamental da Camada Física NB-IoT - Uplink

Behavioral Modeling of the NB-IoT Uplink Physical Layer

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações, realizada sob a orientação científica do Professor Doutor Arnaldo Silva Rodrigues de Oliveira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Professor José Manuel Neto Vieira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Telmo Reis Cunha
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Marco Alexandre Cravo Gomes
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade de Coimbra (Arguente)

Professor Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar da Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

Quero deixar aqui um agradecimento muito especial aos meus pais pelo apoio incondicional dado ao longo destes anos.

Um obrigado ao meu orientador, Professor Doutor Arnaldo Oliveira, e coorientador, Professor Doutor José Neto Vieira, que contribuiram activamente para o sucesso desta dissertação. Agradeço também ao Instituto de Telecomunicações de Aveiro pelas condições de trabalho proporcionadas.

Por último, a todos os amigos pelo apoio, motivação e momentos de alegria proporcionados ao longo deste percurso.

A todos, muito obrigada!

Palavras-chave IoT, NB-IoT, LPWAN, SC-FDMA, turbo coding, equalizer, USRP, front-end

Resumo

A Internet das Coisas (IoT) consiste numa rede sem fios de sensores/atuadores ligados entre si e que têm a capacidade de recolher dados. Devido ao crescimento rápido do mercado IoT, as redes de longa distância e baixa potência (LPWAN) tornaram-se populares. O *NarrowBand-IoT* (NB-IoT), desenvolvido pela *3rd Generation Partnership Project* (3GPP), é um desses protocolos.

O principal objectivo desta dissertação é a implementação de uma simulação comportamental em MATLAB do NB-IoT no uplink, que será disponibilizada abertamente. Esta será focada, primariamente, na camada física e nas suas respetivas funcionalidades, nomeadamente *turbo coding*, modulação SC-FDMA, modelos de simulação de canal, desmodulação SC-FDMA, estimação de canal, equalizador e *turbo decoding*. A estimação de canal é feita usando símbolos piloto previamente conhecidos. Os modelos de canal utilizados são baseados nas especificações oficiais da 3GPP.

A taxa de bits errados (BER) é calculada e usada de forma a avaliar a performance do *turbo encoder* e do equalizador *zero forcing* (ZF). Serve também como comparação quando a implementação usa esquemas de modulação diferentes (*Binary Phase-Shift Keying* (BPSK) e *Quadrature Phase-Shift Keying* (QPSK)). Além disso, os sinais gerados em MATLAB são transmitidos usando como *front-end* de rádio-frequência (RF) uma *Universal Software Radio Peripheral* (USRP). Posteriormente, são recebidos, desmodulados e descodificados. Finalmente, é obtida a constelação do sinal, a BER é calculada e os resultados são analisados.

Keywords IoT, NB-IoT, LPWAN, SC-FDMA, turbo coding, equalizer, USRP, front-end

Abstract

The Internet of Things (IoT) refers to a wireless network of interconnected sensors/actuators with data-collecting technologies. Low Power Wide Area Networks (LPWAN) have become popular due to the rapid growth of the IoT market. Narrowband-IoT (NB-IoT), developed by 3rd Generation Partnership Project (3GPP), is one of these protocols.

The main objective of this thesis is the implementation of an open-source uplink behavioral simulator based on MATLAB. Its focus is primarily on Layer 1 (physical layer) relevant functionalities, namely turbo coding, Single-Carrier Frequency-Division Multiple Access (SC-FDMA) modulation, channel modeling, SC-FDMA demodulation, channel estimation, equalization and turbo decoding. Channel estimation is performed using known pilot symbols. The used channel models are based on the 3GPP official release specs.

The Bit Error Rate (BER) is calculated in order to evaluate the turbo encoder and the Zero Forcing (ZF) equalizer performance, and to compare Binary Phase-Shift Keying (BPSK) and Quadrature Phase-Shift Keying (QPSK) implementations. Furthermore, the MATLAB generated signal is transmitted using a radio-frequency (RF) front-end consisting of an Universal Software Radio Peripheral (USRP). Afterwards, the signal is received, demodulated and decoded. A constellation is obtained, the BER is calculated and the results are analyzed.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project ... 2
ACK/NACK	Acknowledgement/Negative-Acknowledgement ... 51
ADC	Analog-to-Digital Converter ... 67
AWGN	Additive White Gaussian Noise ... vii
BER	Bit Error Ratio ... 70
BPSK	Binary Phase-Shift Keying ... 11
CAZAC	Constant Amplitude, Zero AutoCorrelation ... 31
CCDF	Complementary Cumulative Distribution Function ... 88
CFO	Carrier Frequency Offset ... 4
CP	Cyclic Prefix ... 10
CRC	Cyclic Redundancy Check ... 17
DCI	Downlink Control Information ... 45
DFT	Discrete Fourier Transform ... 12
DL-SCH	Downlink Shared Channel ... 46
DMRS	Demodulation Reference Signal ... 4
DSP	Digital Signal Processing ... 67
eNodeB	Evolved Node B ... 2
EPA	Extended Pedestrian A ... 59
ETU	Extended Typical Urban ... 59
EVA	Extended Vehicular A ... 59
FDD	Frequency Division Duplex ... 9
FFT	Fast Fourier Transform ... 9
FPGA	Field-Programmable Gate Array ... 67
FSM	Finite State Machine ... 20
GSM	Global System for Mobile communications ... 7
HARQ-ACK	HARQ Acknowledgement ... 28
IDFT	Inverse Discrete Fourier Transform ... 34
IFFT	Inverse Fast Fourier Transform ... 12

IoT	Internet of Things	1
ISI	Inter Symbol Interference	27
LoRa	Long Range Radio	2
LPWAN	Low Power Wide Area Networks	1
LTE	Long Term Evolution	3
MAC	Media Access Control	13
MAP	Maximum A Posteriori	38
MATLAB	MATrix LABoratory	3
MMSE	Minimum Mean-Squared Error	99
MSb	Most Significant bit	45
NB-IoT	Narrowband Internet of Things	2
NPDCCH	Narrowband Physical Downlink Control Channel	43
NPDSCH	Narrowband Physical Downlink Shared Channel	43
NPRACH	Narrowband Physical Random Access Channel	13
NPSS	Narrowband Primary Synchronization Signal	44
NPUSCH	Narrowband Physical Uplink Shared Channel	12
NSSS	Narrowband Secondary Synchronization Signal	44
OFDMA	Orthogonal Frequency Division Multiple Access	11
PAPR	Peak-to-Average Power Ratio	12
PRACH	Physical Random Access Channel	17
PUCCH	Physical Uplink Control Channel	14
QoS	Quality of Service	2
QPSK	Quadrature Phase-Shift Keying	11
RACH	Random Access Channel	4
RAR	Random Access Response	14
RF	Radio Frequency	3
RLC	Radio Link Control	13
RNTI	Radio Network Temporary Identifier	46
RSC	Recursive Systematic Convolutional	19
RU	Resource Unit	4
SC-FDMA	Single-Carrier Frequency-Division Multiple Access	9
SDR	Software Defined Radio	67
SIB	System Information Block	44
SIMO	Single-Input and Multiple-Output	8
SINR	Signal-to-Interference-plus-Noise Ratio	25
SISO	Single-Input and Single-Output	8

SMA SubMiniature version A 67
SNR Signal-to-Noise Ratio 84
SOVA Soft Output Viterbi Algorithm 38
STO Symbol Time Offset 4
TBS Transport Block Size 13
TDD Time Division Duplex 9
UCI Uplink Control Information 4
UE User Equipment 2
UHD Universal Hardware Driver 68
UL-SCH Uplink Shared Channel 4
USB Universal Serial Bus 67
USRP Universal Software Radio Peripheral 4
ZC Zadoff-Chu 31
ZF Zero Forcing 41

List of Symbols

rv_{idx}	transmission redundancy version	23
r_u	value of the demodulation reference signals	29

Chapter 1

Introduction

1.1 Framework

Over the last years the Internet of Things (IoT) market has grown exponentially, becoming one of the most active areas of technology development. It is constantly changing and adapting, being the focus of many research and investment initiatives.

The IoT refers to a network of interconnected devices such as sensors/actuators, outfitted with data-collecting technologies in order to communicate with one another. This should enable seamless integration of potentially any object into the Internet, thus allowing for new forms of interactions between human beings and devices, or directly between devices [CVZZ16].

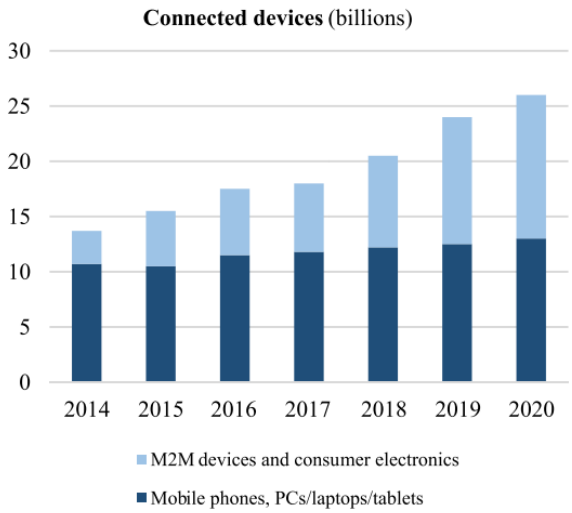


Figure 1.1: Projected number of IoT devices by year [SWH17].

According to Figure 1.1, by 2020, more than twenty billion devices will be connected through wireless communications. Low Power Wide Area Networks (LPWAN) technologies have become popular, in order to fulfill the IoT market's rapid growth. Their shared features are wide coverage, low bandwidth, small data sizes and long battery life. LPWAN protocols are divided in two groups: the ones that use licensed spectrum and the ones that use

unlicensed spectrum. A comparison between two of the leading technologies in each group, 3rd Generation Partnership Project (3GPP) Narrowband Internet of Things (NB-IoT) and Semtech Long Range Radio (LoRa), is presented in Table 1.1.

Table 1.1: NB-IoT and LoRa comparison [SWH17].

	LoRa	**NB-IoT**
Quality of Service	LoRa uses an unlicensed spectrum and cannot offer the same QoS as NB-IoT.	NB-IoT uses a licensed spectrum and its time slotted synchronous protocol is optimal for QoS.
Cost	Spectrum cost is free. Gateways are cheap when compared with NB-IoT base stations.	Spectrum cost is very high. Base stations are more expensive than the LoRa equivalent.
Battery life	LoRa devices can sleep for as little or as long as the application desires, since it is an asynchronous protocol.	In NB-IoT, because of infrequent but regular synchronization, the device consumes additional battery energy.
Latency	LoRa has less energy demands, which leads to a higher latency and lower data rate.	Although the energy demands are bigger in NB-IoT, this leads to a lower latency and higher data rate.
Network coverage	LoRa can cover a whole city using one gateway.	The deployment of NB-IoT is limited to 4G/LTE base stations.
Range	<15 km	<35 km
Deployment model	LoRa components are production-ready now.	Requires a software upgrade for the base stations.
Standard	Closed	Open

In summary, it is shown that LoRa has advantages in terms of battery lifetime and cost, and NB-IoT has benefits regarding Quality of Service (QoS), latency and reliability.

Although both have qualities, there are two key factors worth mentioning. First, LoRa is already fully tested and functional, with modules easily available. Meanwhile, NB-IoT is still to be implemented, requiring a software upgrade in the network's Evolved Node B (eNodeB). Thus, it requires a proof of concept before its hardware implementation, making it the ideal choice for a behavioral model simulation. Secondly, NB-IoT is an open standard, meaning there is enough information available to simulate and analyze this protocol in different platforms. Therefore, NB-IoT will be the main focus of this master thesis.

1.1.1 NB-IoT Overview

This subsection briefly summarizes the NB-IoT uplink and downlink transmission directions, introducing concepts required to further understanding the protocol. Figure 1.2 depicts a transmission between two User Equipments (UEs) and an eNodeB in both directions (uplink and downlink).

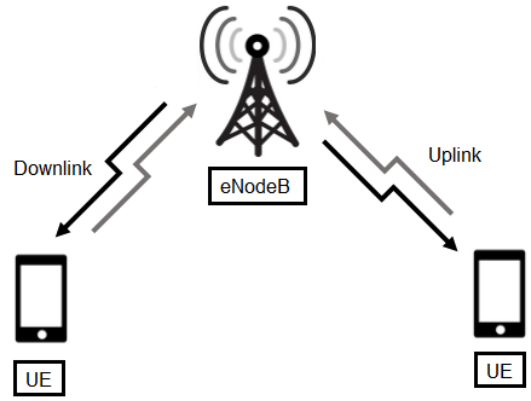


Figure 1.2: Uplink and downlink transmission.

Generally, **uplink** means the UE transmits a signal to the eNodeB, and **downlink** corresponds to a transmission from the eNodeB to the UE.

Each transmission mode requires a huge amount of signal processing, being quite complex. Therefore, two different dissertations were proposed, one for the uplink and one for the downlink, with the uplink transmission being the main focus of this dissertation.

1.2 Motivation

Newly introduced systems, modulation techniques and protocols usually use behavioral models simulated in MATrix LABoratory (MATLAB) to test their functionalities. Implementations without the concern for real time data processing, clock cycles and latency are essential, as they provide a simplified analysis which can be used as a proof of concept. NB-IoT is a new protocol, with several differences when compared to Long Term Evolution (LTE). Therefore, an analysis and implementation of its data generation algorithms is necessary. The simulation results may also be used as a reference for hardware performance tests. Furthermore, it can be integrated in the MATLAB LTE toolbox, which up to this moment doesn't have any NB-IoT uplink functionalities.

A big advantage of these systems is their portability. Besides tests in a simulation environment, it is possible to analyze and verify the transmitter/receiver chain in a co-simulation environment, with the signal being generated/captured using two Radio Frequency (RF) front-ends. This decreases the required amount of time for the development process, reducing the number of bugs that will be found.

1.3 Objectives

The goals of this master thesis are the following:

- Model and simulate the physical layer of a NB-IoT uplink transmitter in MATLAB. This includes all the physical channels and signals with their respective coding and modulation.

- Model and simulate the physical layer of a NB-IoT uplink receiver in MATLAB. This includes the decoding and demodulation of all uplink physical channels and signals.
- Simulate different channel models to test how the generated transmitted signal is affected. Implement and validate a channel estimator and equalizer in order to mitigate those effects.
- Choose a RF front-end compatible with MATLAB, to send and capture signals based on a co-simulation environment. Implement functions that allow for Carrier Frequency Offset (CFO) and Symbol Time Offset (STO) estimation and correction.

1.4 Contributions

The main contributions of this master thesis include:

- The design of a behavioral model for NB-IoT physical layer in MATLAB, including all channels and signals.
- A performance evaluation using different channel models in a simulation environment and two Universal Software Radio Peripherals (USRPs) as RF front-ends, for a closer to reality environment.

1.5 Thesis Structure

This thesis is divided into 8 chapters:

- **Chapter 1 - "Introduction":** contains a brief description about IoT and the overall motivation behind the development and creation of a NB-IoT uplink transceiver. It also briefly summarizes the main goals to be achieved with this master thesis.
- **Chapter 2 - "NB-IoT General Concepts":** presents an overview on the basics of a NB-IoT system, its deployments options, frame structure and duplex arrangements. It's also explained, in detail, the new time-frequency grid and a new concept not present in LTE - Resource Units (RUs). To finish, an overview of all physical channels and signals is done.
- **Chapter 3 - "NB-IoT Uplink Transmitter":** explains the Uplink Shared Channel (UL-SCH) coding (CRC addition, turbo coding and rate matching) and the Uplink Control Information (UCI) channel coding. Furthermore, their modulation (scrambling, modulation mapper and SC-FDMA modulation) is explained in detail. Afterwards, the Random Access Channel (RACH) preamble generation is described, as is the creation of Demodulation Reference Signal (DMRS).
- **Chapter 4 - "NB-IoT Uplink Receiver":** explains the UL-SCH decoding (CRC removal, turbo decoding and rate dematching) and the UCI channel decoding. Furthermore, their demodulation (descrambling, modulation demapping and SC-FDMA demodulation) is explained in detail. Afterwards, the RACH preamble detection is described. Channel estimation and equalization based on the received DMRS is also clarified.

- **Chapter 5 - "NB-IoT Physical Layer Procedures":** explains the required procedures to send data in each specific channel. It is emphasized the scheduling order and how necessary parameters are obtained, in order to generate a transport block to be sent over any channel.
- **Chapter 6 - "MATLAB Simulation and USRP Implementation":** presents block diagrams with the MATLAB simulation and the USRP co-simulation work flow. Its main functions are explained in detail, showing their input and output parameters, with a explanation of their main purpose. It's also justified the reason why MATLAB was chosen.
- **Chapter 7 - "Performance Results":** shows all the results, simulated and measured, throughout this work. It includes constellations, eye diagrams and BER performances. They are compared and discussed to offer a better perspective of the overall developed work.
- **Chapter 8 - "Conclusion and Future Work":** marks the end of this thesis with a summary of the developed work and presents some propositions for possible future works.

The following appendixes were included:

- **Appendix A - "User Guide":** provides a small user guide on how to run the implemented simulations. Furthermore, a step by step explanation on how to test the co-simulation environment is provided.
- **Appendix B - "Reference Sequence Test":** shows what happens when a know sequence is imposed in the beginning of the simulation, contributing with a reference sequence test in each main point of the transmitter and receiver chain for both NPUSCH format 1 and 2.

Chapter 2

NB-IoT General Concepts

This chapter provides introductory concepts, necessary to understand the NB-IoT physical layer.

2.1 Deployment Options

NB-IoT can operate in three different modes [Roh16]:

- Stand alone operation, where a possible option is the replacement of the Global System for Mobile communications (GSM) carriers by a NB-IoT cell. Considering NB-IoT has a bandwidth of 180kHz (discussed in section 2.3) and GSM has a bandwidth of 200 kHz, there is a guard interval of 10 kHz on both sides of the spectrum.
- Guard band operation utilizing resource blocks within a LTE carrier's guard-band.
- In-band operation utilizing one resource block in the LTE carrier.

These different deployment scenarios are illustrated in Figure 2.1.

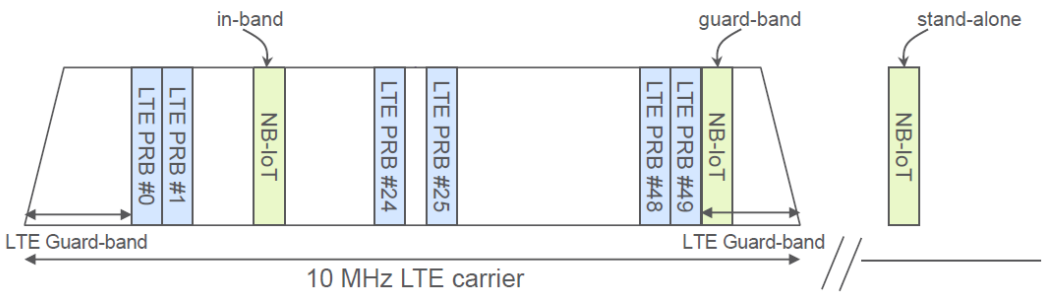
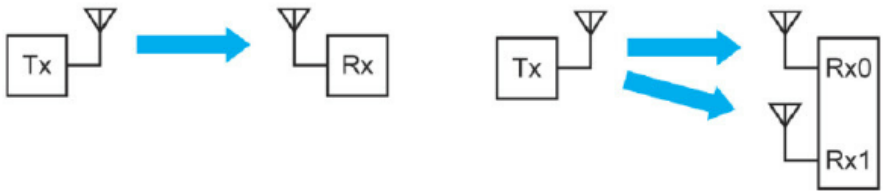


Figure 2.1: Examples of NB-IoT deployments: stand-alone, LTE in-band and LTE guard-band [YPEWR17].

2.2 Antenna Selection

Up to this moment, NB-IoT uplink supports only one transmission mode, which consists on a transport block transmission using a single physical antenna. A wireless communications system in which one antenna is used at the source (transmitter) and one antenna is used at the destination (receiver) is called Single-Input and Single-Output (SISO), being the simplest antenna technology - Figure 2.2a.

In some environments, SISO systems are vulnerable to problems caused by multipath effects. When the waves have obstructions in their path, they are scattered and take different paths to reach the receiver. Each scattered signal arrives at different times, which causes problems, such as fading. In order to minimize them, it is possible to have more than one antenna at the receiver (eNodeB). A system in which one antenna is used at the source (transmitter) and two antennas are used at the destination (receiver) is called Single-Input and Multiple-Output (SIMO) - Figure 2.2b [Blo17b].



(a) Antenna configuration for SISO systems.

(b) Antenna configuration for SIMO systems.

Figure 2.2: Antenna configuration for SISO and SIMO systems [LGV14].

2.3 Subcarrier Spacing

In the uplink, both 3.75kHz and 15kHz subcarrier spacing (Δf) are supported. Observing Table 2.1, it's possible to conclude that when Δf is equal to 15kHz the number of uplink subcarriers (N_{sc}^{UL}) is 12 and when Δf equals 3.75kHz the N_{sc}^{UL} is 48. Thus, the bandwith for NB-IoT is easily obtained multiplying Δf by N_{sc}^{UL} - 15kHz $\times$ 12 = 3.75kHz $\times$ 48 = 180kHz.

Table 2.1: Number of subcarriers, frame length, subframe length and slot length values depending on Δf.

Parameters	Δf **= 15kHz**	Δf **= 3.75kHz**
Number of subcarriers	12	48
Radio frame length	10ms	40ms
Subframe length	1ms	4ms
Slot length	0.5ms	2ms

If Δf is equal to 15kHz, the radio frame has a length of T_f = 10ms, with subframes of length T_{sf} = 1ms and slots of length T_{slot} = 0.5ms. The symbol time is $1/\Delta f \approx 66.7\mu$s. If Δf has the value of 3.75kHz, the radio frame has a length of T_f = 40ms, with subframes of length T_{sf} = 4ms and slots of length T_{slot} = 2ms. The symbol time is $1/\Delta f \approx 266,7\mu$s.

2.4 Frame Structure

In LTE, two types of frame structures are supported, selected according to the duplex mode: Time Division Duplex (TDD) or Frequency Division Duplex (FDD). Since in NB-IoT, only FDD mode is supported (discussed in section 2.5), frame structure type 1 is used.

2.4.1 Frame Structure Type 1

In the time domain, NB-IoT transmissions are organized into radio frames, each of which is divided into ten equal subframes, as illustrated in Figure 2.3. Each subframe consists of two equal slots, with each slot consisting of seven symbols, including cyclic prefix. Each symbol is Single-Carrier Frequency-Division Multiple Access (SC-FDMA) modulated - discussed in section 2.7.1.

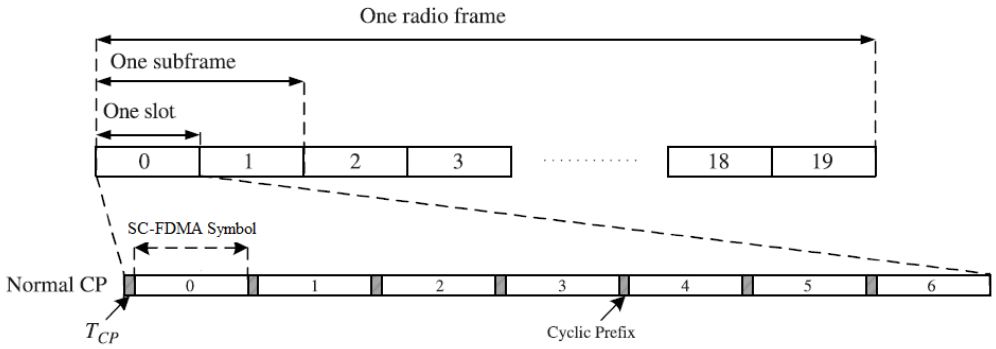


Figure 2.3: Frame structure type 1 [GZAM10].

To provide consistent and exact timing definitions, different time intervals within the NB-IoT specifications are defined as multiples of a basic time unit $Ts = 1/(15000 \times 128)$s when $\Delta f = 15$kHz or $Ts = 1/(3750 \times 512)$s when $\Delta f = 3.75$kHz. The basic time unit Ts can thus be seen as the sampling time of a transmitter/receiver with a Fast Fourier Transform (FFT) size equal to 128 when $\Delta f = 15$kHz or 512 when $\Delta f = 3.75$kHz.

2.5 Duplex Arrangements

Although in LTE both TDD and FDD are supported (providing spectrum flexibility), in NB-IoT only FDD half-duplex type B mode is supported.

2.5.1 Frequency-Division Duplex

In case of FDD operation, uplink and downlink use different carrier frequencies, denoted f_{UL} and f_{DL}. In each frame, there are ten uplink subframes and ten downlink subframes, and uplink and downlink can happen simultaneously. Isolation between both transmissions is obtained using duplex filters. In certain frequency bands with a very narrow duplex gap, it is challenging to design the duplex filters. In this case, a device only supports half-duplex operation. Half-duplex operation requires a guard period where the device can switch between transmission and reception.

LTE supports two ways of providing the necessary guard period [DPS16]:

- Half-duplex type A, where guard time for the switch is handled by setting the necessary amount of timing advance in the devices.
- Half-duplex type B, where a whole subframe is used as guard between reception and transmission, with an oscillator that is retuned between uplink and downlink frequencies.

Thus, for NB-IoT, uplink and downlink are separated in frequency and the UE either receives or transmits, but never does both simultaneously. In addition, between every switch from uplink to downlink or vice-versa there is at least one guard subframe in between, allowing the UE to switch its transmitter to receiver chain and vice-versa (Figure 2.4). This duplex scheme requires less complex hardware, allowing a lower-cost implementation, ideal for IoT applications.

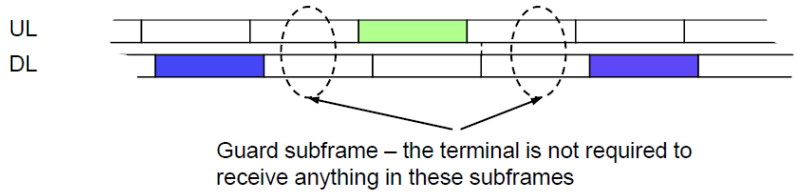


Figure 2.4: Guard time for half-duplex FDD type B duplex scheme [DPS16].

2.6 NB-IoT Time-Frequency Grid

This section describes the basic NB-IoT time-frequency transmission grid. Contrary to LTE where two Cyclic Prefix (CP) lengths are defined, NB-IoT only supports the normal CP length, corresponding to seven symbols per slot. Figure 2.5 depicts the resource grid for a single slot.

When Δf is equal to 15kHz, the resource grid is equal to the one for LTE with normal CP, using only one resource block. When Δf is equal to 3.75kHz, the resource grid for a slot has a different structure, since the N_{sc}^{UL} is 48, instead of 12.

A resource element, which is the smallest physical resource in NB-IoT, is indicated in Figure 2.5 by one square. Furthermore, resource elements are grouped into a resource block. If $\Delta f = 15$kHz, each resource block consists of 12 consecutive subcarriers in the frequency domain ($N_{sc}^{UL} = 12$) and one 0.5ms slot in the time domain. Therefore, each resource block consists of $7 \times 12 = 84$ resource elements. If $\Delta f = 3.75$kHz, each resource block consists of 48 consecutive subcarriers in the frequency domain ($N_{sc}^{UL} = 48$) and one 2ms slot in the time domain. Each resource block thus consists of $7 \times 48 = 336$ resource elements. Each resource element in the resource grid is defined by the index pair (k, l) in a slot, where $k = 0, ..., N_{sc}^{UL} - 1$ and $l = 0, ..., N_{symb}^{UL} - 1$ are the indices in the frequency and time domain, respectively [DPS16].

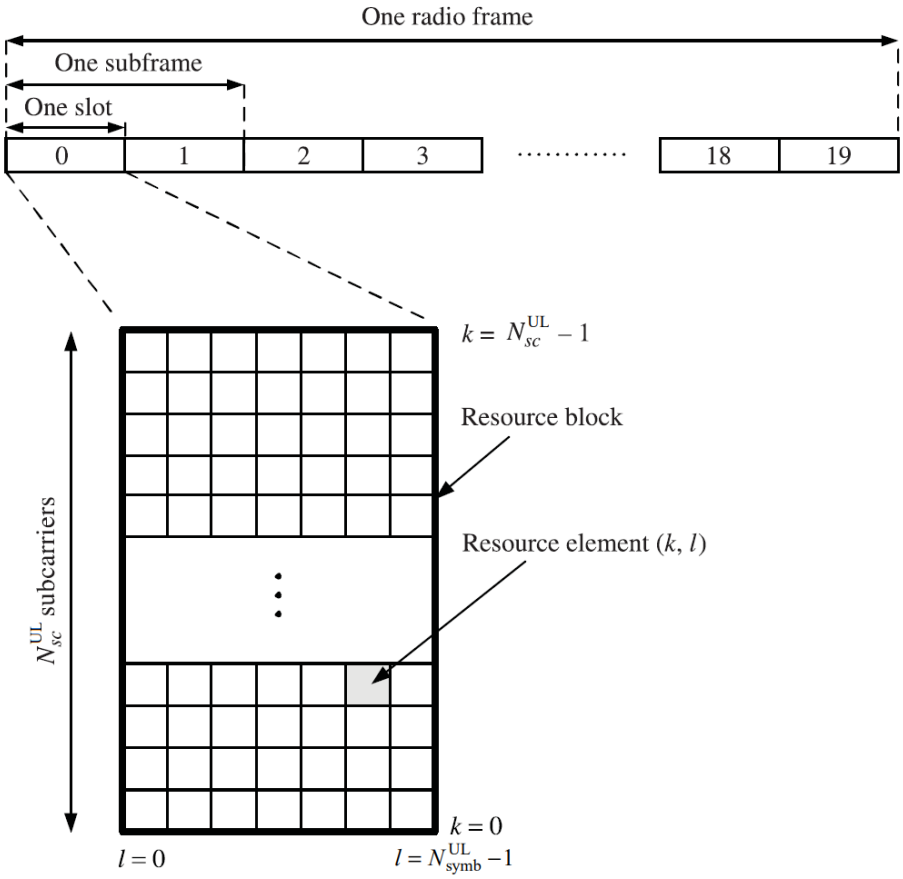


Figure 2.5: Structure of the uplink resource grid [GZAM10].

2.7 Transmission Scheme

The NB-IoT uplink uses both multi-tone and single-tone transmissions. Multi-tone transmission uses $\Delta f = 15$kHz, equal to LTE. While only the 12-tone format is supported by LTE UEs, 6-tone and 3-tone formats are added on NB-IoT UEs. Single-tone transmission is introduced for NB-IoT and supports two Δf values, 15kHz and 3.75kHz [YPEWR17]. Uplink transmissions are based on SC-FDMA modulation.

2.7.1 SC-FDMA Overview

A graphical comparison of Orthogonal Frequency Division Multiple Access (OFDMA) (used in downlink) and SC-FDMA is shown in Figure 2.6. Although NB-IoT signals are allocated in 12 or 48 adjacent subcarriers ($N_{sc}^{UL} = 12$ or 48), only four subcarriers are depicted. Even though uplink NB-IoT accepts both Binary Phase-Shift Keying (BPSK) and Quadrature Phase-Shift Keying (QPSK) modulation schemes, in this example, data is represented using QPSK modulation.

In the OFDMA example of Figure 2.6, in each symbol period, four QPSK symbols are inserted in parallel, one per subcarrier. After each symbol period, the CP is inserted and the next four symbols are transmitted. Although the CP is shown as a gap, it is actually a copy of

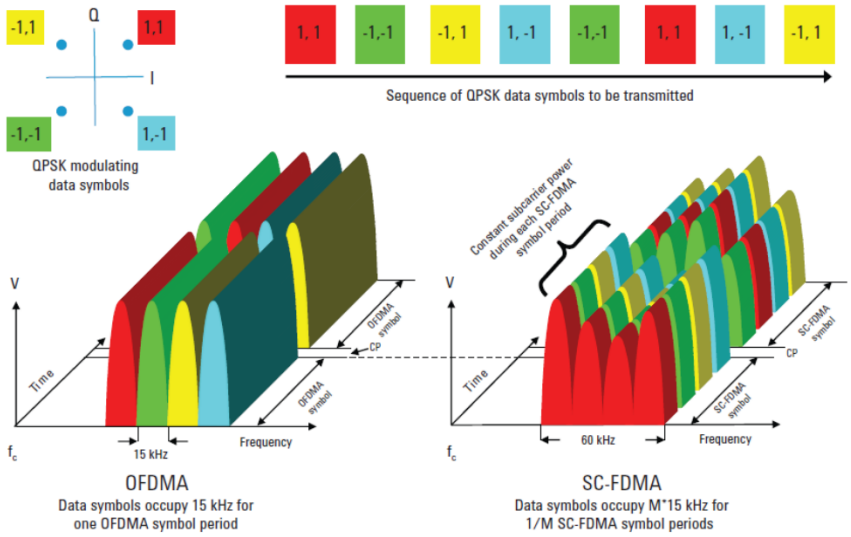


Figure 2.6: Comparison of OFDMA and SC-FDMA multiple access technologies [Not09].

the next symbol's ending. To create the transmitted signal, an Inverse Fast Fourier Transform (IFFT) is performed on each subcarrier. Adding together many parallel narrowband QPSK waveforms creates an undesirably high Peak-to-Average Power Ratio (PAPR). In the uplink this effect would be particularly damaging, since it would drain the UE battery rapidly.

In the SC-FDMA example of Figure 2.6, the four QPSK data symbols are transmitted in series at four times the rate, with each data symbol occupying $M \times 15$kHz bandwidth. Therefore, the SC-FDMA signal appears to be more like a single-carrier, with each data symbol being represented by one wide signal. SC-FDMA signal generation can be seen as a Discrete Fourier Transform (DFT)-precoded OFDMA, which spreads the information over all the subcarriers. After the DFT, all the other steps occur as in OFDMA.

Detailed information about QPSK and BPSK modulation can be found on sub-subsection 3.1.2.2. More about SC-FDMA signal generation is presented in sub-subsection 3.1.2.3.

2.8 Resource Units

The smallest unit to map a transport block is the RU. A RU is defined as the number of uplink slots in each resource unit (N_{slots}^{UL}) multiplied by the number of consecutive subcarriers in a resource unit (N_{sc}^{RU}), while in the frequency domain it is simply given by the total number of resource units (N_{RU}) parameter. In NB-IoT, the number of symbols in a slot ($N_{symbols}^{UL}$) is always 7.

The chosen RU depends on the Narrowband Physical Uplink Shared Channel (NPUSCH) format and Δf. For NPUSCH format 1, there are five options presented in Table 2.2.

For NPUSCH format 2, the RU is always composed of one subcarrier with a length of four slots (Table 2.3).

Table 2.2: RU duration for NPUSCH format 1.

Δf	N_{sc}^{RU}	N_{slots}^{UL}	**RU duration**
3.75kHz	1	16	32ms
15kHz	1	16	8ms
	3	8	4ms
	6	4	2ms
	12	2	1ms

Table 2.3: RU duration for NPUSCH format 2.

Δf	N_{sc}^{RU}	N_{slots}^{UL}	**RU duration**
3.75kHz	1	4	8ms
15kHz	1	4	2ms

2.9 NB-IoT Hierarchical Channel Structure and Reference Signals

In this section, it is described the NB-IoT hierarchical channel structure. There are three different channel types: logical channels, transport channels, and physical channels. Logical channels carry data and signaling messages between the Media Access Control (MAC) and Radio Link Control (RLC) layers. Transport channels provide data characteristics between the MAC and physical layers, such as the modulation scheme. Physical channels are the implementation of transport channels over the radio interface, with a number of resource elements of the time-frequency grid carrying information from higher layers. Besides physical channels, there is a signal embedded in the uplink physical layer, which does not carry information. This signal is called DMRS.

Figures 2.7 and 2.8 specify, respectively, the mapping of the UL-SCH and the RACH to their corresponding physical channels: NPUSCH format 1 and Narrowband Physical Random Access Channel (NPRACH). Figure 2.9 specifies the mapping of the UCI to its corresponding physical channel (NPUSCH format 2).

2.9.1 Uplink Physical Channels

Since this master thesis is mainly focused on the physical layer, an overview of the different physical channels and signal is done in this subsection.

2.9.1.1 NPUSCH Format 1 with DMRS

NPUSCH format 1 carries UL-SCH data, with maximum Transport Block Size (TBS) value of 1000 bits (much smaller than LTE). NPUSCH format 1 supports both multi-tone and single-tone transmissions using 7 symbols per slot ($N_{symbols}^{UL} = 7$), with the middle symbol being used to allocate the DMRS [Roh16, YPEWR17]. Further details regarding coding and modulation are discussed in section 3.1. The DMRS generation is detailed in section 3.3.

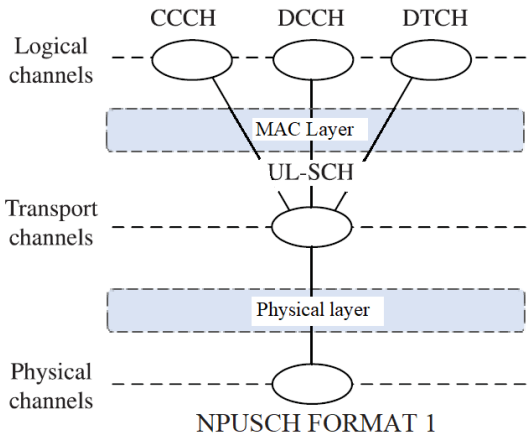


Figure 2.7: Uplink shared channel mapping [GZAM10].

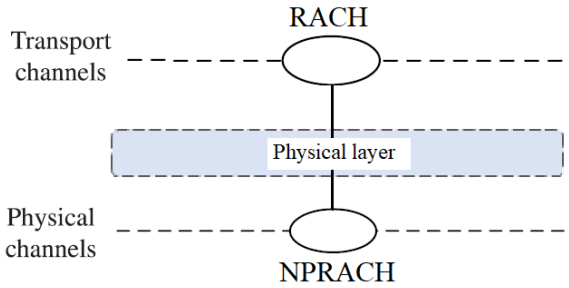


Figure 2.8: Random access channel mapping [GZAM10].

2.9.1.2 NPUSCH Format 2 with DMRS

NPUSCH format 2 carries UCI, which is restricted to an acknowledgment of a downlink transmission. Even though in a downlink transmission it is configurable whether a transmission shall be acknowledged, on uplink there is always an acknowledgment when a downlink transmission in received. NPUSCH format 2 also has 7 symbols per slot ($N^{UL}_{symbols} = 7$), but uses the middle three symbols to allocate the DMRS [Roh16, YPEWR17]. Further details regarding coding and modulation are discussed in section 3.2. The DMRS generation is detailed in section 3.3.

2.9.1.3 NPRACH

NPRACH carries the random access preamble, whose signal generation is explained in section 3.4. It is used to iniciate the Random Access Response (RAR) procedure, when sent by the UE.

In summary, all data is sent over the NPUSCH, except for RACH transmission. This includes also the UCI, which is transmitted using a different format. Consequently, there is no equivalent to the Physical Uplink Control Channel (PUCCH) in LTE [Roh16, GZAM10].

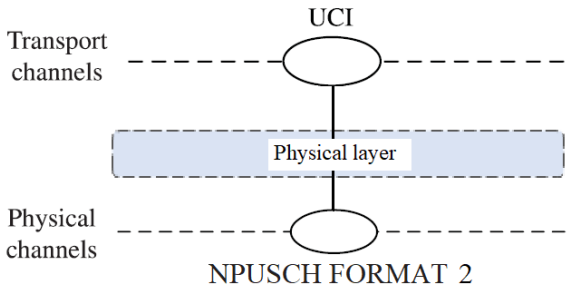


Figure 2.9: Uplink control information mapping [GZAM10].

In this chapter, NB-IoT introductory concepts were explained. Deployment options, frame structure and duplex arrangements were summarized. The used time-frequency grid was depicted, and a new concept not used in LTE was introduced - RUs. A brief introduction to the available channels and signals was made. In the next chapter, the uplink transmitter architecture for all available physical channels will be explained in detail. Furthermore, the DMRS generation will be described.

Chapter 3

NB-IoT Uplink Transmitter

This chapter provides detailed information about the NB-IoT uplink physical layer transmitter. Section 3.1 describes the physical layer processing applied to the UL-SCH, section 3.2 explains the physical layer processing applied to the UCI, section 3.3 provides information about the DMRS and section 3.4 gives detailed information about the Physical Random Access Channel (PRACH) preamble generation. Knowledge about the transmitter processing chain helps to fully understand the overall system operation implemented on MATLAB.

3.1 Uplink Shared Channel

This section describes the physical-layer processing applied to the UL-SCH and the subsequent mapping to the uplink physical resource in the form of the basic time-frequency grid. Figure 3.1 outlines the different steps of the UL-SCH physical layer processing.

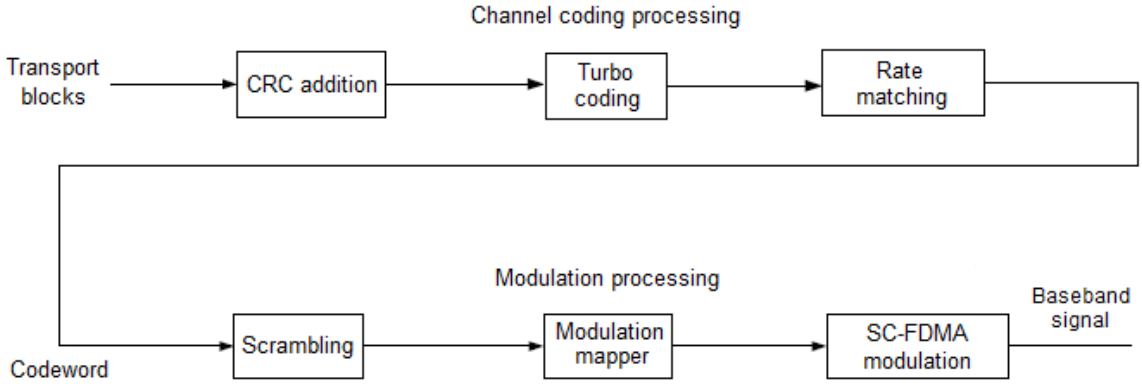


Figure 3.1: Block diagram of the NB-IoT UL-SCH transmitter [GZAM10].

It is divided in coding and modulation procedures. The process called coding assists on the recovery of the transport block at the receiver side. Cyclic Redundancy Check (CRC) addition helps with error-detection, turbo coding adds redundancy to the data by adding two new data streams, which will be affected differently in case of a burst error. Finally, rate matching spreads out the occurrence of errors and punctures bits to fit the payload size. The process called modulation performs bit scrambling using a Gold sequence, modulates the

bits according to a specific modulation scheme and generates a baseband signal, where each symbol is based on SC-FDMA. Subsections 3.1.1 and 3.1.2 describe channel coding procedures and modulation procedures, respectively.

3.1.1 Channel Coding Processing

This subsection describes channel coding procedures. The process called channel coding transforms into codewords user data coming from transport blocks. This procedure, helps to correct/detect errors caused by distortion during transmission. The several coding procedures and their necessary steps are going to be explained in detail.

3.1.1.1 CRC Addition

CRC codes are an error-detection technique and the first step performed during the coding process. First, the transport block is divided by a generator polynomial. According to section 5.2.2.1 of [3GP17a], the 24A generator type is the baseline for the UL-SCH (Table 3.1).

Table 3.1: Polynomial used on CRC addition in NB-IoT.

Type	**Generator polynomial**
24A	$D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$

Then, the previous division remainder is appended to the transport block. An example can be seen in Figure 3.2.

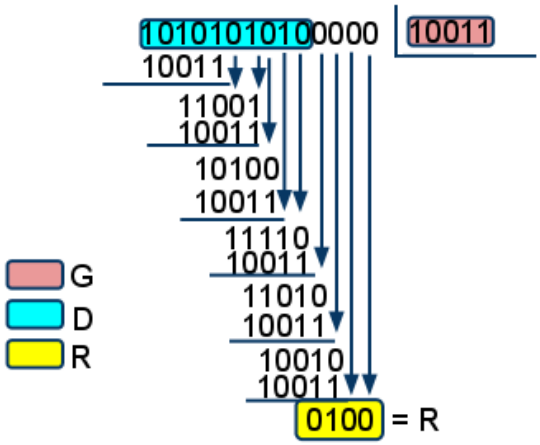


Figure 3.2: CRC addition example [Blo17a].

Let's denote the polynomial generator by G and the transport block by D. In the example, $G = 10011$ and $D = 1010101010$. Since G is 5 bits long, then the remainder's length (denoted by r) is $G - 1 = 4$, $r = 4$. D is shifted left by r bits and zeros will be inserted into those places. The new pattern will be denoted by $D' = 10101010100000$. Now D' is divided by G, using an "exclusive OR" operation. The remainder from the division, R, will be concatenated with the transport block D [Blo17a].

Considering the generator polynomial length is 25, the remainder added to the transport block will always have length equal to 24. Therefore, the output length (denoted by K) after the CRC addition will be $K = TBS + 24$.

3.1.1.2 Turbo Coding

This sub-subsection describes the turbo encoder and its components in detail. The turbo encoder is built using two identical Recursive Systematic Convolutional (RSC) encoders in parallel. The two RSC are separated by an interleaver, with its output being a permuted version of the input data. The purpose of the interleaver is to provide some degree of de-correlation among the inputs of each encoder. Therefore, in the event of a burst error, the two code streams are affected differently, allowing data to still be recovered. The structure of the turbo encoder is illustrated in Figure 3.3.

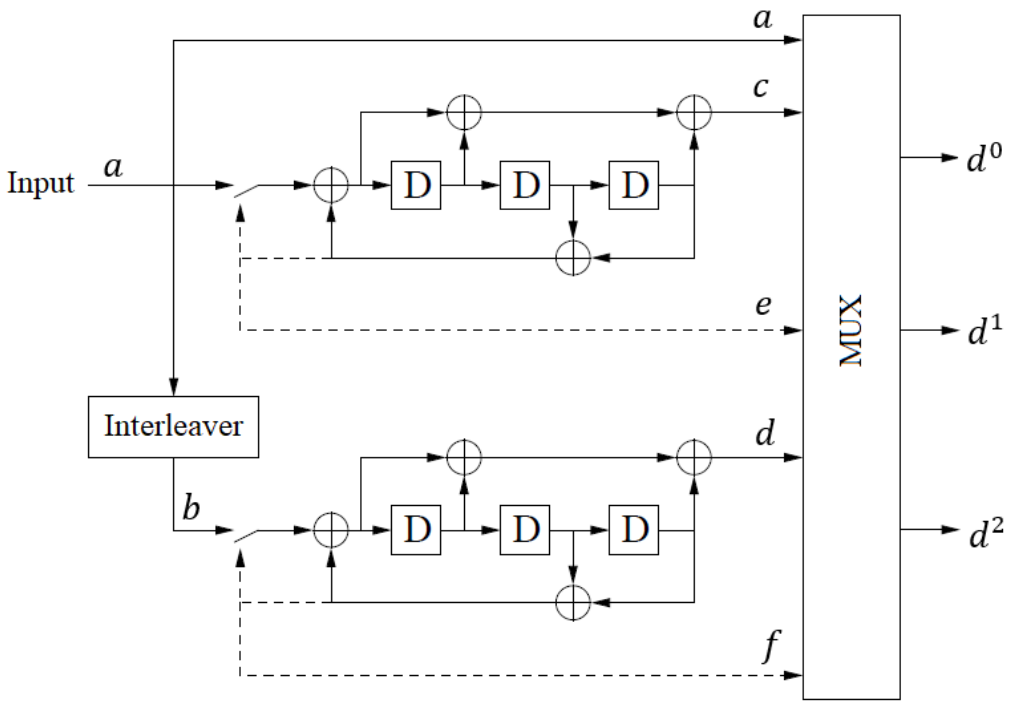


Figure 3.3: Turbo encoder block diagram (dotted lines apply for trellis termination only) [3GP17a].

At the beginning of encoding a message block, the RSC encoders are in a defined state of all-zeros. Three bit-streams are produced, the systematic signal (a), and one output of each encoder (parity signals c and d). The interleaved payload signal (b) is not transmitted, except during the trellis termination phase, as it can be easily reconstructed at the receiver side from a. After all the data bits have been encoded, trellis termination is performed bringing the encoders to an all-zeros state once again. To achieve this, the switches in Figure 3.3 are moved in the down position. The input, in this case, is shown by dashed lines. It takes three bits to force the final state back to all-zero state for each encoder. The output bit stream includes not only the tail bits corresponding to the upper encoder (e), but also the

tail bits corresponding to the lower encoder (f). In addition to these six termination bits, six corresponding parity bits for the upper and lower encoder are also transmitted. Finally, all the bits are multiplexed into three data streams (d^0, d^1 and d^2), that correspond to the input of the next step (rate-matching). Considering K the input length, the total length of the encoded bit sequence becomes $3K+12$, where each stream has $outsize = K+4$ bits. The encoder code rate is calculated in equation 3.1.

$$r = \frac{K}{3K+12} \tag{3.1}$$

However, for a large size of K, the loss in code rate due to tail bits is negligible and thus, with the code rate being approximately 1/3 [Imr13, Joo10].

RSC encoder: Each RSC encoder, has three memory bits forming an 8-state Finite State Machine (FSM). To understand the operation of the FSM, the state transition can be shown as a trellis diagram in Figure 3.4. Variable a is the input sequence and c is the output sequence. S_1, S_2 and S_3 are the current state of the three memory bits in the encoder. S_1^+, S_2^+ and S_3^+ are the next state of the three memory bits. $State_1$ to $State_8$ correspond to the eight possible states. Figure 3.4 shows all the possible transitions for the encoding bits in a sequence. The first state in a transition is always the all-zeros one, corresponding to $State_1$.

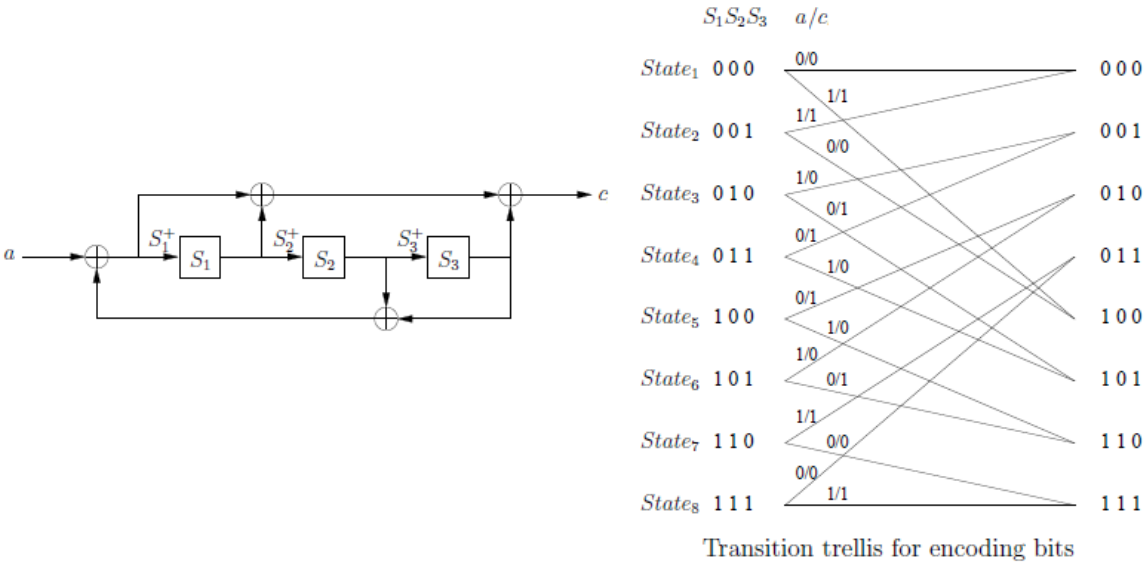


Figure 3.4: Scheme of the convolutional encoder and corresponding trellis diagram for the encoding bits [Li09].

The transition of the states and the decoding results can be expressed as the following equations:

$$S_1^+ = a_k \oplus S_2 \oplus S_3 \tag{3.2}$$
$$S_2^+ = S_1 \tag{3.3}$$
$$S_3^+ = S_2 \tag{3.4}$$
$$c_k = S_1^+ \oplus S_1 \oplus S_3 \tag{3.5}$$

Trellis termination: As mentioned above, this method involves three extra bits at the end of each sequence to force the encoder return to the all-zeros state. These extra bits are also sent to the decoder. In Figure 3.5, the state transition can be shown as a trellis diagram for the termination bits. Variable e is the input sequence and c is the output sequence. S_1, S_2 and S_3 are the current state of the three memory bits in the encoder. S_1^+, S_2^+ and S_3^+ are the next state of the three memory bits. With this type of termination technique, the last state in the sequence is forced back to $State_1$. This causes a number of limited transitions.

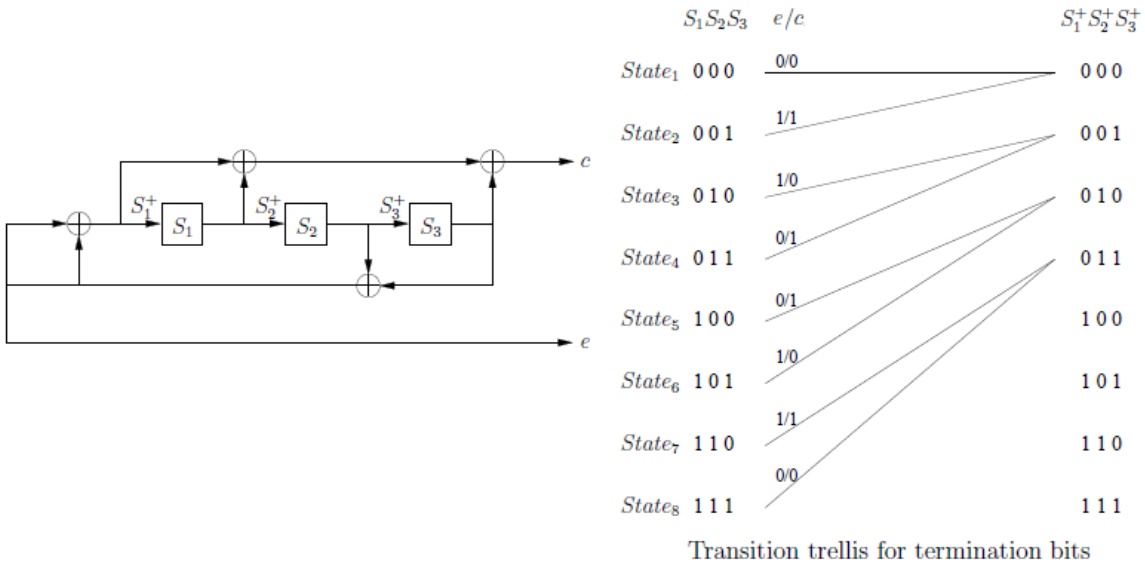


Figure 3.5: Scheme of the convolutional encoder and corresponding trellis diagram for the termination bits [Li09].

The transition of the states and the decoding results can be expressed as the following equations:

$$S_1^+ = 0 \tag{3.6}$$
$$S_2^+ = S_1 \tag{3.7}$$
$$S_3^+ = S_2 \tag{3.8}$$
$$e_k = S_2 \oplus S_3 \tag{3.9}$$
$$c_k = 0 \oplus s_1 \oplus S_3 \tag{3.10}$$

3.1.1.3 Rate Matching

The rate matching step forms a transport block that fits the payload size, calculated using the modulation order (Q_m) and the total number of resource units (N_{RU}). There are three steps that compose the rate matching process. Namely, sub-block interleaver, bit collection and bit selection (puncturing) (Figure 3.6).

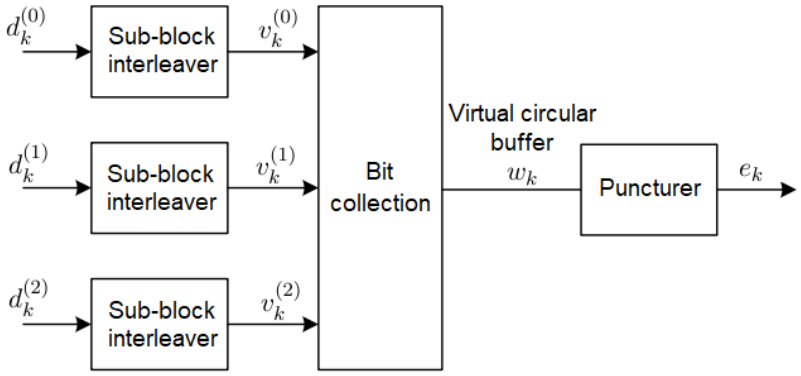


Figure 3.6: Rate matching block diagram [3GP17a].

Interleaving is performed in order to spread out the occurrence of burst errors, which improves the overall performance of the decoder on the receiver side. Since the interleaving is performed separately for the systematic and parity bits obtained at the output of the turbo encoder, a bit collection stage is required to place the three bit streams in a specific order. Finally, the bit selection (puncturing) stage is needed in order to puncture some of the bits to create the required payload [Mat17c, GZAM10].

Sub-block interleaving: The inputs to the three sub-block interleavers correspond to each output stream from the turbo coding step. The interleaving is performed independently for each bit stream, using inter-column permutations.

Each input stream (with length D), is arranged into a matrix having C columns, $C = 32$. The number of rows, R, is determined in such a way that $C \times R >= D$.

If D is not divisible by 32, it is necessary to add bits of nulls at the beginning of the matrix so it is completely full. The number of nulls, N, is $N = C \times R - D$. The matrix size is $K = N + D$;

After, the matrix is rearranged using the inter-column permutation specified in Table 3.2.

Table 3.2: Inter-column permutation pattern for sub-block interleaver.

Number of columns (C)	Inter-column permutation pattern
32	<0, 16, 8, 24, 4, 20, 12, 28, 2, 18, 10, 26, 6, 22, 14, 30, 1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31>

Bit collection: The bits collected from the interleaved streams are rearranged, forming a virtual circular buffer. The systematic bits are placed at the beginning, followed by the two interleaved parity streams, which are bit-by-bit interlaced, as shown in Figure 3.7. This

assures that an equal number of both versions of parity bits are transmitted. A virtual circular buffer is then formed and denoted by W_k, of size $3K$ bits [GZAM10].

Puncturer: The bit selection extracts consecutive bits from the circular buffer so that it fits into its assigned physical resource unit. To select the output bit sequence, its length should first be determined. Using the length value, denoted by E, bits are read from the virtual circular buffer. The starting point, denoted by k_0 of the bit selection depends on the transmission redundancy version (rv_{idx}).

The calculation of the total number of bits available for the transmission of one transport block (E) and the starting point (k_0) values goes as follows:

1) Calculate Nc, which denotes the soft buffer size for the current transport block: $Nc = Kw = 3 \times K$ (only for NB-IoT uplink).

2) Obtain G value, which is a parameter given by superior layers of the network and depends on the modulation order (Q_m) and the total number of resource units (N_{RU}).

3) For NB-IoT uplink, $E = G$.

4) Calculate k_0, which is $k_0 = R \times ((2 \times \lceil Nc/(8 \times R) \rceil \times rv_{idx}) + 2)$, where rv_{idx} is the redundancy version given to the UE by the eNodeB - subsection 5.1. For NB-IoT uplink, rv_{idx} can have the value zero or two.

5) While bypassing the null bits added in the sub-block interleaver step, and using the output sequence length (E) and the starting point for the bit collection (k_0), a codeword is obtained (Figure 3.7).

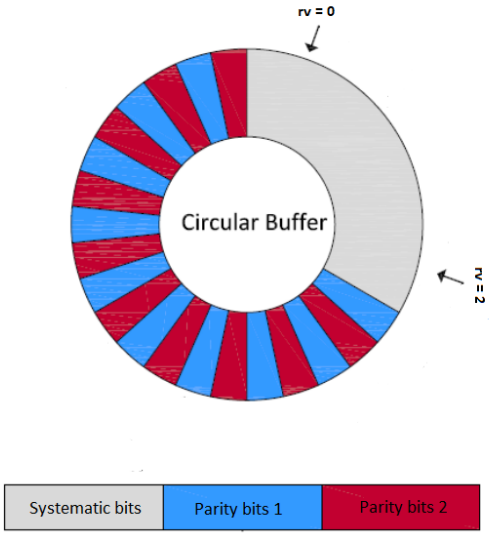


Figure 3.7: Virtual circular buffer with the three interleaved encoded bit streams. The arrows represent the starting point for the bit selection depending on the rv_{idx} value [SP16].

After the rate matching process, a codeword is obtained. The coding procedure is finished.

3.1.2 Modulation Processing

This subsection describes the generic modulation procedures. In NB-IoT, modulation takes one codeword and converts it to a complex-valued baseband signal. As shown in Figure

3.1, the modulation processing consists of scrambling, modulation mapping and SC-FDMA modulation.

3.1.2.1 Scrambling

The first step on the modulation processing chain is the scrambling of coded data, which randomizes interference and avoids long sequences of equal bits.

The codeword obtained after the coding process (section 3.1.1) is scrambled according to equation 3.11:

$$\tilde{b}(i) = (b(i) + c(i)) \bmod 2,\ i = 0, 1, \ldots, M_b \tag{3.11}$$

where $b(i)$ denotes the codeword of length M_b to be scrambled and $c(i)$ denotes the pseudo-random Gold sequence described below.

Gold sequence: The Gold sequence is a result of modulo-2 binary addition of two sequences given by equations 3.12, 3.13 and 3.14:

$$x1(n + 31) = (x1(n + 3) + x1(n)) \bmod 2 \tag{3.12}$$
$$x2(n + 31) = (x2(n + 3) + x2(n + 2) + x2(n + 1) + x2(n)) \bmod 2 \tag{3.13}$$
$$c(n) = (x1(n + Nc) + x2(n + Nc)) \bmod 2 \tag{3.14}$$

The value of Nc is equal to 1600 and the length n is chosen as needed. Instead of the sequences being randomly generated, they are selected, making them only pseudo-random.

As can be seen in equations 3.12 and 3.13, there are 30 undefined values. The first sequence ($x1$) is always initialized with $x1(0) = 1$, $x1(n) = 0$, $n = 1, 2, 3, \ldots, 30$. The second sequence ($x2$) initialization is uniquely assigned depending on the Gold sequence application.

For this specific application, $x2$ is initialized with c_{init} (equation 3.15) after its conversion to binary.

$$c_{init} = \text{RNTI} + n_f \bmod 2.2^{13} + \lfloor n_s/2 \rfloor .2^9 + N_{ID}^{Ncell} \tag{3.15}$$

where RNTI is the radio network temporary identifier and N_{ID}^{Ncell} the narrowband physical layer cell identifier. The frame number (n_f) and the slot number (n_s) vary with each transmission.

3.1.2.2 Modulation Mapper

The second step on the modulation processing chain consists on modulation mapping. For uplink, the supported data modulation schemes in NB-IoT include QPSK and BPSK, whose choice depends on the selected number of number of consecutive subcarriers in a resource unit (N_{sc}^{RU}):

- For RUs with one subcarrier, BPSK and QPSK may be used.
- For all other RUs, QPSK is applied.

The block of scrambled bits with length M_b is modulated into a block of complex-valued symbols, where M_s is the total number of modulated symbols. The relation between M_s and M_b is defined in equation 3.16.

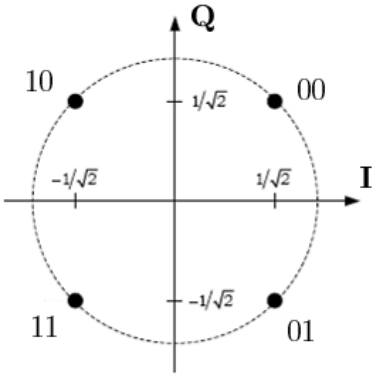
$$M_s = \frac{M_b}{Qm} \tag{3.16}$$

The modulation order (Q_m) represents the number of bits in the modulation scheme constellation. The relationship between the modulation scheme used and Q_m is represented on Table 3.3 [GZAM10].

Table 3.3: Relationship between the modulation scheme and Q_m.

Qm	**Modulation scheme**
1	BPSK
2	QPSK

QPSK: It is a modulation technique that uses 2 bits per symbol. The modulation mapping is done according to Figure 3.8. There are four states (four possible combinations using two bits, $2^2 = 4$). The theoretical bandwidth efficiency is two bits/second/Hz.



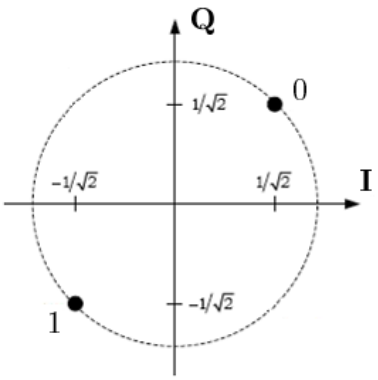
b(i), b(i+1)	**I**	**Q**
00	$1/\sqrt{2}$	$1/\sqrt{2}$
01	$1/\sqrt{2}$	$-1/\sqrt{2}$
10	$-1/\sqrt{2}$	$1/\sqrt{2}$
11	$-1/\sqrt{2}$	$-1/\sqrt{2}$

Figure 3.8: QPSK modulation mapping.

This modulation mapping uses Gray coding, which means that constellation points that are closer to each other differ in as few bits as possible. Therefore, fewer bits will be wrong, if the decoding is done incorrectly.

BPSK: It is a modulation technique that uses 1 bit per symbol. The modulation mapping is done according to Figure 3.9. There are two states (two possible combinations using one bit, $2^1 = 2$). The theoretical bandwidth efficiency is one bit/second/Hz. BPSK is regarded as the most robust digital modulation technique and is used for long distance wireless communication.

Q_m is selected based on the measured Signal-to-Interference-plus-Noise Ratio (SINR). Each modulation scheme has a threshold SINR. UEs closer to the eNodeB (with higher SINR values) use less robust modulation schemes. Meanwhile, UEs located further from the eNodeB (with lower SINR values) use a more robust modulation scheme. The eNodeB always selects the modulation order (Q_m) to be used in uplink transmissions [Tel15].



b(i)	I	Q
0	$1/\sqrt{2}$	$1/\sqrt{2}$
1	$-1/\sqrt{2}$	$-1/\sqrt{2}$

Figure 3.9: BPSK modulation mapping.

3.1.2.3 SC-FDMA Modulation

Although NB-IoT uses OFDMA in the downlink, the uplink utilizes SC-FDMA. This is due to the fact that the overall value of PAPR is smaller then in OFDMA, for all modulation schemes. Therefore, it will consume considerably less energy, which is a fundamental characteristic for the UE operation.

SC-FDMA is divided in several steps, namely DFT precoding, resource mapping, padding addition, IFFT, resampling and CP addition. The schematic of the SC-FDMA modulation is represented in Figure 3.10.

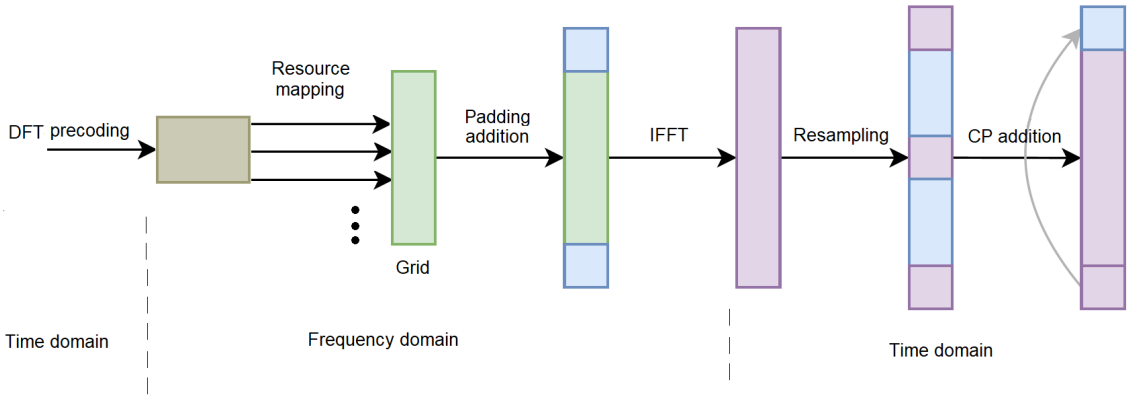


Figure 3.10: SC-FDMA modulation schematic [DT17].

DFT precoding: SC-FDMA is a DFT coded OFDMA, which means that before going through the standard OFDMA modulation, time domain data symbols are converted to frequency domain using a DFT [RBB17].

Resource Mapping: The resource mapping of each complex-valued symbol onto its corresponding resource element is done in increasing order, beginning with subcarriers and then SC-FDMA symbols, while bypassing DMRS. In Figure 3.11, the available resource unit has 6 subcarriers, therefore the resource mapping would go through 3 more slots - Table 2.2. Each slot is then repeated a certain number of times according to the parameter N_{Rep} - section 5.4.

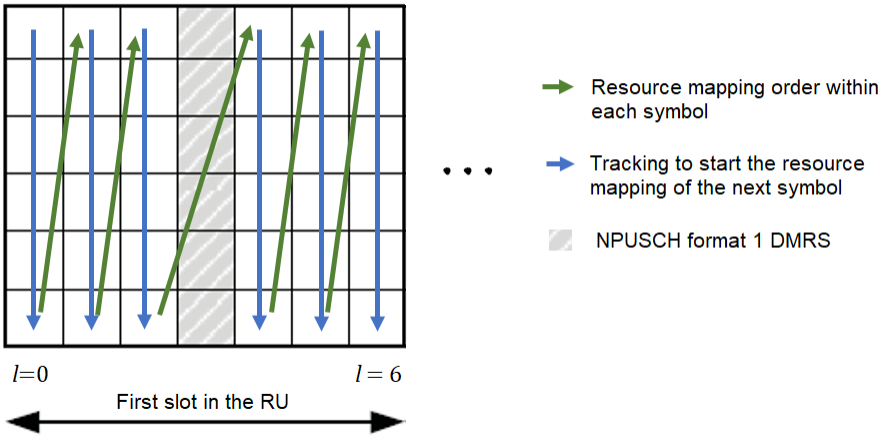


Figure 3.11: Resource mapping using a resource unit with six available subcarriers.

Padding addition: A symbol consists of 12 or 48 resource elements, depending on Δf. Each symbol is padded with additional zeros at each end, so it increases its size to the next power of two. This step has two goals. First, by being a power of two, the next step is simplified - IFFT. Secondly, if the DFT and IFFT had equal size, one would annul the other's effect.

IFFT: The baseband signal is obtained using an IFFT operation. An IFFT transforms complex frequency domain symbols into a time domain signal.

Resample: Considering $F_s = \frac{1}{T_s} \Leftrightarrow F_s = 1.92\text{MHz}$, the time domain signal needs to be resampled, so a higher sampling rate of 1.92MHz is obtained. This is done, using interpolation, by a factor of eight.

CP addition: The term CP refers to the prefixing of each SC-FDMA symbol with a repetition of its end (Figure 3.12). The prefixing size varies with two factors: if it is the first symbol in the slot or not and according to the Δf used.

The main objective of the CP addition is to be used as a guard interval, which eliminates the Inter Symbol Interference (ISI) [GZAM10, Roh16, DT17].

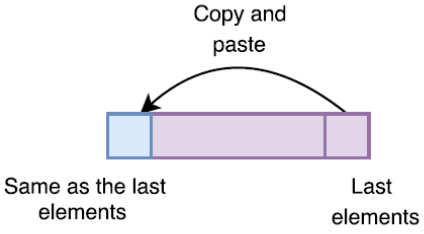


Figure 3.12: Cyclic prefix addition [DT17].

When the SC-FDMA modulation is terminated,the physical layer processing applied to the UL-SCH on the transmitter side is finished. The signal is in the baseband format.

3.2 Uplink Control Information

This section describes the physical-layer processing applied to the UCI and its mapping in the time-frequency grid. Figure 3.13 outlines the different steps of the UCI physical layer processing. It is divided in coding and modulation procedures, with subsections 3.2.1 and 3.2.2 describing channel coding procedures and modulation procedures, respectively.

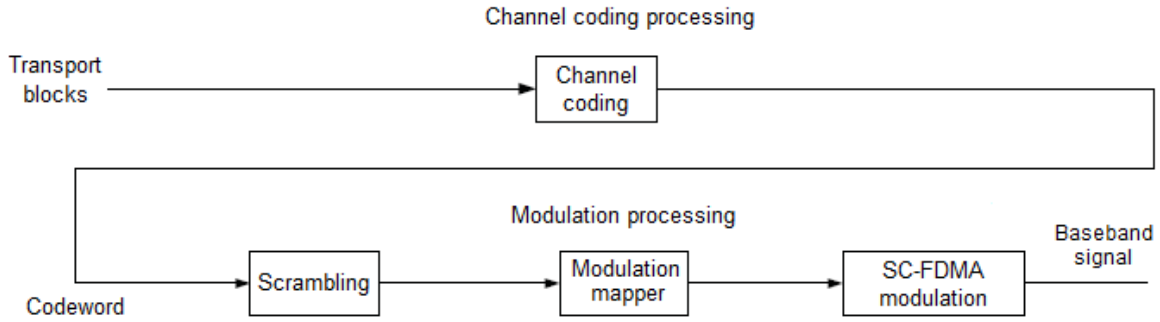


Figure 3.13: Block diagram of the NB-IoT UCI transmitter [GZAM10].

3.2.1 Channel Coding Processing

This subsection describes the only channel coding step that transforms control information data into a codeword.

Control data is only sent on the NPUSCH format 2 when there is no UL-SCH data. The control data arrives to the channel coding unit in the form of an HARQ Acknowledgement (HARQ-ACK) indicator. The one bit information of HARQ-ACK is coded according to Table 3.4.

Table 3.4: HARQ-ACK codewords.

HARQ-ACK	HARQ-ACK codeword
0	<0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
1	<1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1>

After this coding process, a codeword is obtained.

3.2.2 Modulation Processing

Modulation Processing steps are performed in exactly the same way as for the UL-SCH. Scrambling is done according to sub-subsection 3.1.2.1. Modulation mapping is done according to sub-subsection 3.1.2.2, always using BPSK as a modulation scheme. SC-FDMA modulation is done according to sub-subsection 3.1.2.3 always using a single subcarrier and, therefore, a single-tone transmission. When this is terminated, the physical layer processing applied to the UCI on the transmitter side is finished. The signal is in baseband format.

3.3 Demodulation Reference Signals

The value of the demodulation reference signals (r_u) is transmitted on uplink resource units assigned to the UE and used for coherent demodulation/detection of data and control information at the eNodeB.

If the number of consecutive subcarriers in a resource unit (N_{sc}^{RU}) is bigger than one, DMRS symbols are constructed from a base sequence multiplied by a phase factor defined in equation 3.17.

$$r_u(n) = e^{j(\varphi(n)\pi/4+\alpha n)} \tag{3.17}$$

where the value of $\varphi(n)$ depends on the number of uplink subcarriers (N_{sc}^{UL}) and the parameter u. If group hopping is not enabled, u is given by a higher layer parameter. If this parameter is undefined, then it is equal to a fixed value depending on N_{ID}^{Ncell} and the N_{sc}^{RU}.

If the N_{sc}^{RU} equals one, DMRS symbols are constructed based on equations 3.18, 3.19 and 3.20.

$$\bar{r_u}(n) = \frac{1}{\sqrt{2}}(1+j)(1-2c(n))w(n \bmod 16) \tag{3.18}$$

$$r_u(n) = \bar{r_u}(n), \text{if NPUSCH format 1} \tag{3.19}$$

$$r_u(3n+m) = \bar{w}(m)\bar{r_u}(n), \text{if NPUSCH format 2} \tag{3.20}$$

where c(n) is a Gold sequence initialized with $c_{init} = 35$ (paragraph 3.1.2.1) and $w(n)$ depends on the variable u.

For NPUSCH format 2, $\bar{w}(m)$ is a spreading orthogonal sequence.

3.3.1 Group Hopping Enabled

It is important to note that group hopping can only be enabled for NPUSCH format 1. If group hopping is enabled, u is defined by a group hopping pattern (f_{gh}) and a sequence-shift pattern (f_{ss}). f_{gh} is based on a Gold sequence (paragraph 3.1.2.1) initialized with $c_{init} = \lfloor \frac{N_{ID}^{Ncell}}{N_{seq}^{RU}} \rfloor$. f_{ss} depends on the N_{ID}^{Ncell}, N_{seq}^{RU} and a higher-layer parameter called Δ_{ss}. N_{seq}^{RU} is a parameter that is based on the N_{sc}^{RU}. If Δ_{ss} is not defined, it assumes the value zero [3GP16, Roh16].

3.3.2 Resource Mapping of Demodulation Reference Signals

The DMRSs are transmitted in either one or three SC-FDMA symbols per slot, depending on the selected NPUSCH format.

For NPUSCH format 1 and $\Delta f = 3.75$kHz, DMRS are transmitted in column number four ($l = 4$). For $\Delta f = 15$kHz, they are transmitted in column number three ($l = 3$). These are the symbols indicated in red in Figure 3.14.

For NPUSCH format 2 and $\Delta f = 3.75$kHz, DMRS are transmitted in columns number zero, one and two ($l = 0, 1, 2$). For $\Delta f = 15$kHz, they are transmitted in columns number two, three and four ($l = 2, 3, 4$). These are the symbols indicated in red in Figure 3.15.

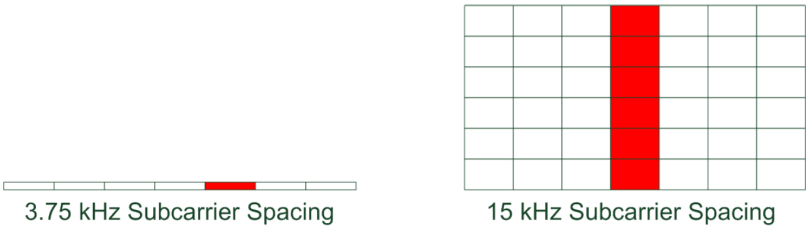


Figure 3.14: Resource elements used for DMRS in NPUSCH format 1. On the left, is an example when Δf is 3.75kHz. On the right, is an example with 6 subcarriers and Δf = 15kHz [Roh16].

Figure 3.15: Resource elements used for demodulation reference signals in NPUSCH format 2. In this format, the RU only occupies one subcarrier [Roh16].

3.4 Random Access Channel

Transmitting a random access preamble is the first step of the RAR procedure (discussed in section 5.3) and allows the UE to establish a connection with the network. This procedure begins when the UE transmits a random access preamble on the NPRACH. The transmitted preamble is based on symbol groups using only one subcarrier. Each symbol group has a CP followed by five symbols. Figure 3.16 shows this sequence.

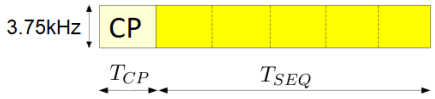


Figure 3.16: Preamble symbol group [Roh16].

Two preamble formats are defined and differ in their CP length. The five symbols have a duration of $T_{SEQ} = 1.333$ms, appended with a CP of $T_{CP} = 67\mu$s for format 0 and 267μs for format 1, giving a total length of 1.4ms and 1.6ms, respectively. In frequency, Δf of 3.75kHz is applied.

The preamble is composed of four symbol groups transmitted continuously. Frequency hopping is applied, with each symbol group being transmitted on a different subcarrier. By construction, hopping is restricted to a set of 12 subcarriers, selected inside the total number of 48. The preamble can be repeated 1, 2, 4, 8, 16, 32, 64, or 128 times by the UE, as indicated by the eNodeB. Figure 3.17 shows an example of a preamble repeated at least four times, where each blue rectangle describes one preamble symbol group.

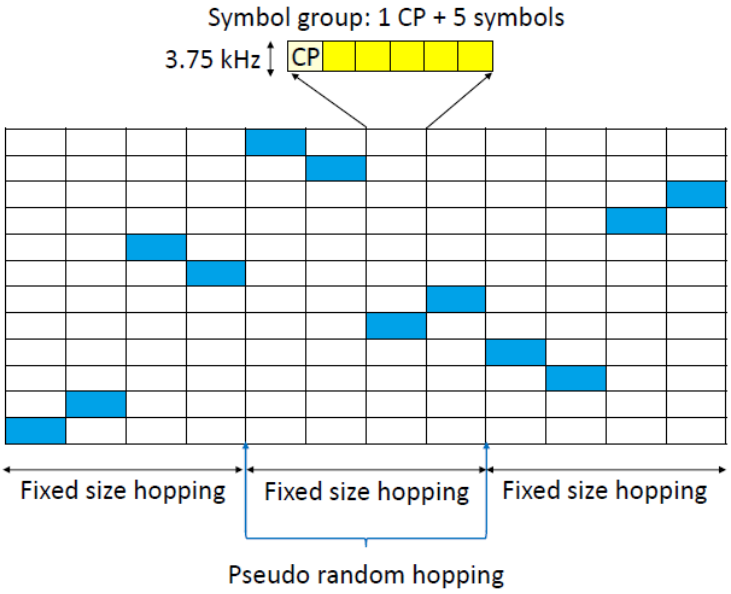


Figure 3.17: NB-IoT PRACH design [LAW16].

3.4.1 Hopping pattern

The hopping design, observed in Figure 3.17, "consists of both inner layer fixed size hopping and outer layer pseudo random hopping. Outer layer pseudo random hopping is applied between four symbol groups. Inner layer fixed size hopping is applied between each symbol group" [LAW16].

The eNodeB selects the subcarrier to be used on the first symbol group transmission. The next three symbol groups are determined by an algorithm which depends only on the location of the first one. For the subcarrier selection of the first symbol group for the next repetition, a pseudo-random hopping is applied, where N_{ID}^{Ncell} and the number of NPRACH repetitions per attempt (N_{rep}^{NPRACH}) are used as input. Again, only this result will influence the subcarrier selection of the following symbol groups [Roh16].

3.4.2 Preamble Generation - Baseband Signal

The preamble symbol group sequence is based on a Zadoff-Chu (ZC) sequence, which depends on the subcarrier location. ZC sequences are a type of Constant Amplitude, Zero AutoCorrelation (CAZAC) sequences. They possess an useful property: their autocorrelation is approximately zero, helping with their posterior detection in the eNodeB.

In this chapter the NB-IoT transmitter chain was presented. The physical layer processing applied to the transport channels (UL-SCH, UCI and RACH), which leads to the generation of a baseband signal, was explained in detail. In the next chapter, the receiver chain will be described. When contraposing both chains, one annuls the others' effect, which allows the transmitted data to be recovered.

Chapter 4

NB-IoT Uplink Receiver

This chapter provides detailed information about the NB-IoT uplink receiver, particularly the physical layer. Section 4.1 describes the reception and subsequent decoding of an UL-SCH transport block, section 4.2 describes the reception and subsequent decoding of an UCI transport block, section 4.3 provides information about the DMRS utility in the receiver side and section 4.4 provides information about the detection of a PRACH preamble coming from the UE.

4.1 UL-SCH Recovery

This section describes the baseband signal demodulation into several codewords and subsequent decoding in order to obtain a recovered transport block in the receiver. On the transmitter, in the SC-FDMA modulation step, each slot is repeated a certain number of times. Therefore, one codeword is obtained for each repetition. Afterwards, the several codewords are decoded and the results are combined, with that combination value being the recovered transport block. Figure 4.1 outlines the different steps to recover a transport block in case of reception on a single antenna. Subsections 4.1.1 and 4.1.2 describe demodulation and decoding procedures, respectively.

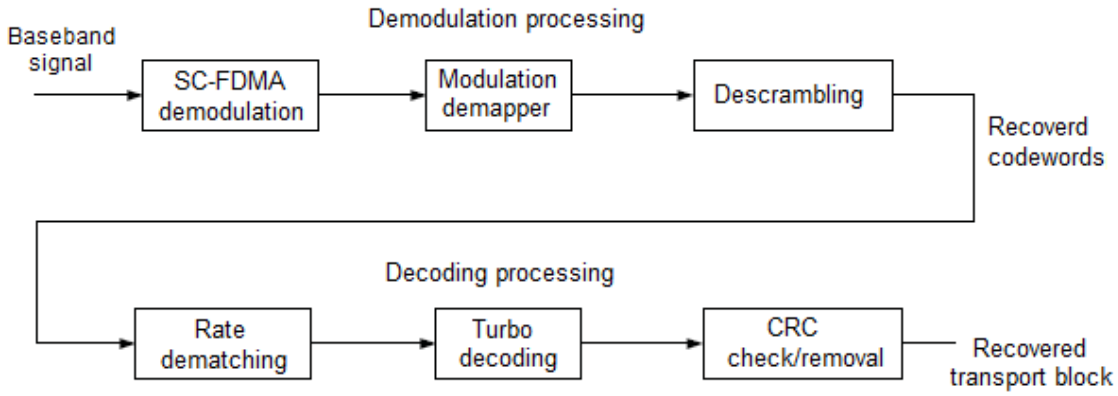


Figure 4.1: Block diagram of the NB-IoT UL-SCH receiver [GZAM10].

4.1.1 Demodulation Processing

This subsection describes the generic demodulation procedures. In NB-IoT, demodulation converts the baseband signal into a certain number of codewords. As shown in Figure 4.1, the demodulation processing consists of SC-FDMA demodulation, modulation demapping and descrambling.

4.1.1.1 SC-FDMA Demodulation

SC-FDMA is divided in several steps, namely, CP removal, resampling, FFT, padding removal, resource demapping and DFT deprecoding. The SC-FDMA demodulation schematic is represented in Figure 4.2.

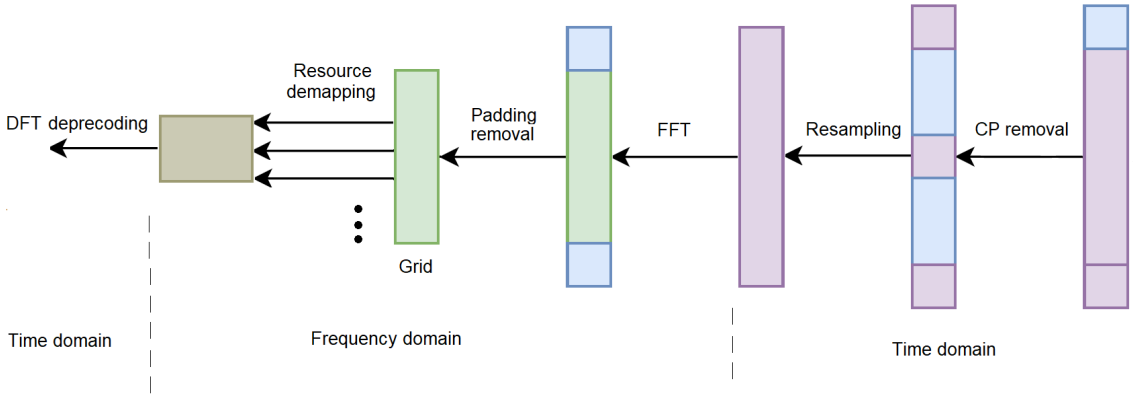


Figure 4.2: SC-FDMA demodulation schematic [DT17].

Cyclic prefix removal: CP removal is done by eliminating the previously added prefix, in the beginning of each symbol.

Resample: The signal is resampled using decimation, by a factor of eight. This way, the original sampling rate is obtained.

FFT: An FFT is performed, so the time domain signal is transformed into the frequency domain.

Padding removal: The elements positioned on the extremities of each symbol are removed, and the data from the middle of the vector is extracted.

Resource Demapping: The symbol value in each resource element is collected, beginning with each subcarrier and then onto SC-FDMA symbols. Since each slot is repeated a certain number of times (discussed in section 5.4), each repetition is demapped and goes on to the next step.

DFT deprecoding: After going through the standard OFDMA demodulation, an Inverse Discrete Fourier Transform (IDFT) is performed, so the frequency domain data symbols are transformed into the time domain [DT17].

4.1.1.2 Modulation Demapper

Each block of complex modulated symbols is demodulated into a block of scrambled bits. Considering M_s the total number of modulated symbols and M_b the total number of demodulated bits, the relationship between them is represented in equation 3.16. The

modulation scheme considered depends on Q_m and their relationship is represented in Table 3.3.

Demodulation mapping can only be done in the soft form, since the bits need to be in that format when they enter the turbo decoder.

QPSK: Modulation demapping for QPSK is done according to Table 4.1. These values don't take into account any type of signal degradation (AWGN, channel fading). Although the demodulation is made in soft decision format, the correspondent value for hard decision is also shown.

Table 4.1: QPSK modulation demapping.

Symbol	**Soft decision**		**Hard decision**	
	b(i)	**b(i+1)**	**b(i)**	**b(i+1)**
$1/\sqrt{2}+(1/\sqrt{2})i$	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0
$1/\sqrt{2}-(1/\sqrt{2})i$	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	1
$-1/\sqrt{2}+(1/\sqrt{2})i$	$-1/\sqrt{2}$	$1/\sqrt{2}$	1	0
$-1/\sqrt{2}-(1/\sqrt{2})i$	$-1/\sqrt{2}$	$-1/\sqrt{2}$	1	1

BPSK: Modulation demapping for BPSK is done according to Table 4.2. The represented values don't take into account any type of signal degradation (AWGN, channel fading). Both soft and hard decision formats are shown.

Table 4.2: BPSK modulation demapping.

Symbol	**Soft decision**	**Hard decision**
$1/\sqrt{2}+(1/\sqrt{2})i$	$1/\sqrt{2}$	0
$-1/\sqrt{2}-(1/\sqrt{2})i$	$-1/\sqrt{2}$	1

4.1.1.3 Descrambling

Let's consider $\tilde{b}(n)$, the bit sequence to be descrambled, is received in the soft decision format. First, a Gold sequence denoted by $c(n)$ is generated using the same initialization values as in the transmitter (sub-subsection 3.1.2.1). The first sequence ($x1$) is initialized with $x1(0)=1$, $x1(n)=0$, $n=1,2,3,\ldots,30$ and the second sequence ($x2$) is initialized with c_{init} after its value is converted to binary (equation 3.15).

Table 4.3: Descrambling operation between the Gold sequence $c(n)$ and the received bits $\tilde{b}(n)$.

$c(n)$	$\tilde{b}(n)$	$b(n)$
0	soft(0)	soft(0)
0	soft(1)	soft(1)
1	soft(0)	soft(1)
1	soft(1)	soft(0)

It is important to note the operation made to obtain $\tilde{b}(n)$ in sub-subsection 3.1.2.1 was a "modulo-2" between the bits to be scrambled $b(n)$ and the Gold sequence $c(n)$. Thus, it's easy to reverse the operation. Observing Table 4.3, it is possible to conclude that when $c(n)$ equals one, the descrambled bit will be the opposite of $\tilde{b}(n)$. When $c(n)$ equals zero, the descrambled bit is equal to $\tilde{b}(n)$.

After all the bits in the vector $\tilde{b}(n)$ are descrambled, a codeword is obtained. The demodulation process is finished.

4.1.2 Channel Decoding Processing

This subsection describes generic channel decoding procedures. The process called decoding transforms a codeword into a transport block. The several decoding procedures and their necessary steps are going to be explained in detail.

4.1.2.1 Rate Dematching

The rate dematching in NB-IoT performs depuncturing and deinterleaving, in order to recover the original output of the turbo encoder. Depicted in Figure 4.3 are several basic steps composing a rate-dematching.

Null positioning: The Transport Block Size (TBS) is known a priori by the eNodeB-section 5.4. Thus, it is possible to calculate the expected output length of the turbo encoder. Taking into account the added CRC will have 24 bits (3.1.1.1), $K = TBS + 24$. Therefore, the output of each stream of the turbo encoder should be $outsize = K + 4$, due to the trellis termination stage.

Reading sub-subsection 3.1.1.3, it is possible to understand that after the nulls are added in the beginning of each matrix (sub-block interleaver stage), they are interleaved according to a specific pattern. The first step would be to position those nulls in their correct place. To do that, three vectors of zeros ($z_k^{(0)}$, $z_k^{(1)}$ and $z_k^{(2)}$) with length $outsize$ are created. Then, they are sub-block interleaved (nulls are added in this stage) - $n_k^{(0)}$, $n_k^{(1)}$ and $n_k^{(2)}$. Afterwards, they are reordered, forming a circular buffer exactly as it was done in the transmitter stage - w_{temp}. The location of the nulls is needed for the depuncturing process.

Depuncturer: The second step would be to depuncter the transport block. In the previous step, it was obtained a circular buffer filled with zeros and nulls positioned in the correct place. In this stage, while bypassing the nulls, known bits are put in their correct position and the unknown bits continue with value zero.

To put bits in their original position, it is necessary to know at what point they started being collected in the original vector. The starting point of the bit selection (k_0) depends on the rv_{idx} of the current transmission. The calculation is done as in sub-subsection 3.1.1.3. At this point, a virtual circular buffer (w_k) is recovered.

Bit separation: A circular buffer is formed by collecting systematic bits at the beginning, followed by bit-by-bit interlacing of the two interleaved parity streams. Knowing this, it is easy to separate the three bit streams (systematic and parity streams) in their original order.

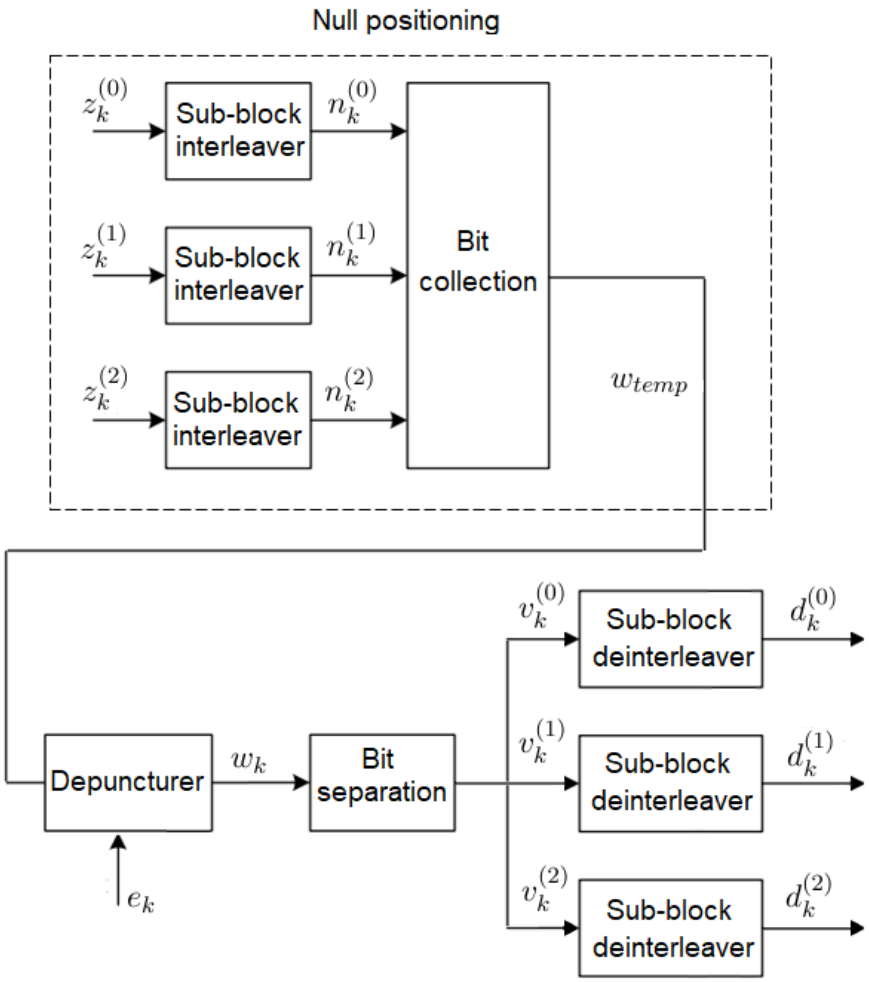


Figure 4.3: Rate dematching block diagram.

Sub-block deinterleaving: The deinterleaving is performed independently for each bit stream, using the same inter-column permutation as the transmitter (Table 3.2).

Each input stream (with length D), is arranged into a matrix having C columns, $C = 32$. The number of rows, R, is determined in such a manner that $C \times R = D$.

After, the matrix is rearranged using the inter-column permutation in order to recover the original column order. Finally, the nulls are removed and the matrix is reshaped into a vector. The three outputs of the turbo encoder/inputs of the rate matching are recovered.

4.1.2.2 Turbo Decoder

The architecture of the turbo decoder is as shown in Figure 4.4. Between the decoder 1 and 2 is formed a loop that performs the iterative decoding process. Since the input of one decoder includes the output of the other decoder, they have to operate alternately. The demultiplexer inputs are in the soft decision format. The five inputs $\tilde{a^c}$, $\tilde{e^c}$, $\tilde{c^c}$, $\tilde{f^c}$ and $\tilde{d^c}$ correspond to the soft format version of the turbo encoder outputs a, e, c, f and d - sub-subsection 3.1.1.2.

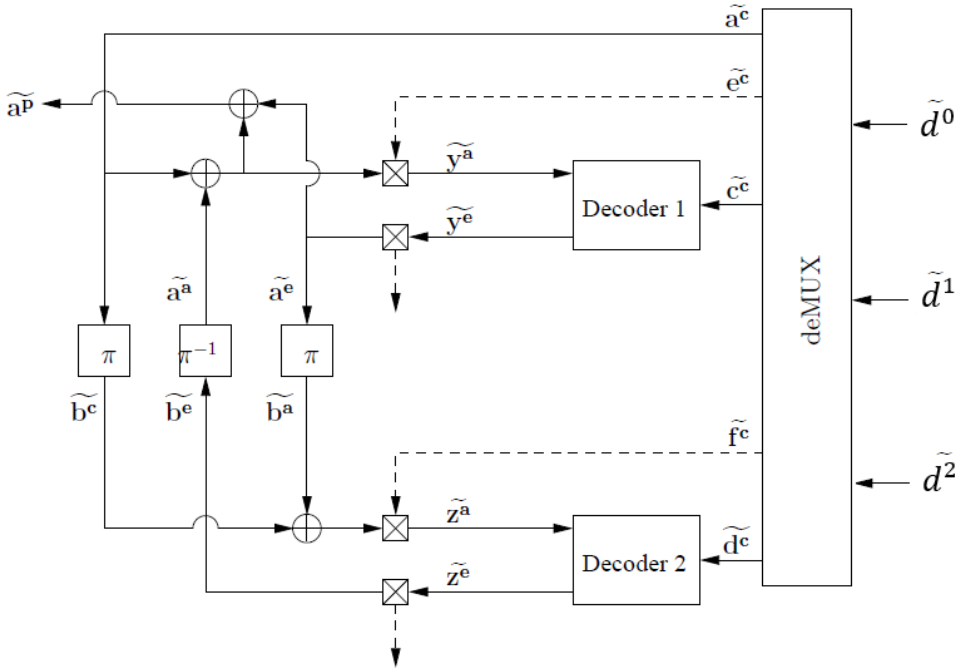


Figure 4.4: Scheme of the NB-IoT turbo decoder [Li09].

Each decoder has as input two sequences. The first is $\tilde{c^c}$ for decoder 1 and $\tilde{d^c}$ for decoder 2, corresponding to the soft decisions of the turbo encoded sequence parity bits. The other one is formed by adding $\tilde{b^a}/\tilde{a^a}$ to the received systematic information. $\tilde{b^a}/\tilde{a^a}$ are generated by the decoders after rearranging their order by the proper interleaver (π) or deinterleaver (π^{-1}). For decoder 1, the input $\tilde{y^a}$ is the sum of $\tilde{a^a}$ and $\tilde{a^c}$ concatenated with $\tilde{e^c}$. For decoder 2, the input $\tilde{z^a}$ is the sum of $\tilde{b^a}$ and interleaved systematic information $\tilde{b^c}$ concatenated with $\tilde{f^c}$.

In the first iteration, $\tilde{b^e}$ is initialized with a sequence of zeros, since its value is completely unknown. Several iterations are performed. According to [Mau10], with its increase, there is an improvement in performance results. After the forth iteration, the increase is rather small, and therefore, does not compensate the required computational resources. Thus, this turbo decoder was implemented using, always, four iterations. Furthermore, with each iteration a CRC check is performed in order to evaluate if there is errors in the transport block. If not, the loop ends and less iterations are required.

Log-MAP algorithm - decoder: There are several algorithms that can be used to implement the turbo decoding decoders. Log-Maximum A Posteriori (MAP) algorithm was chosen. The algorithm offers a trade-off between complexity and error correction performance when compared to Soft Output Viterbi Algorithm (SOVA) (low complexity and error correction performance) and MAP (high complexity and error correction performance) [Li09].

The algorithm is used to generate the decoders' outputs ($\tilde{y^e}$ and $\tilde{z^e}$) and is divided in five parts:

1. For a transition T, γ calculation is done according to equations 4.1 and 4.2:

$$\gamma_y(T) = (1 - y(T))\tilde{y^a}_{n(T)} \tag{4.1}$$

$$\gamma_c(T) = (1 - c(T))\tilde{c^c}_{n(T)} \tag{4.2}$$

2. For a state S, α is calculated according to equation 4.3:

$$\alpha(S) = \max(\gamma_y(T) + \gamma_c(T) + \alpha(fr(T))) \tag{4.3}$$

   where $\alpha(S_1) = 0$ and fr represents a forward recursion.

3. For a state S, β is calculated according to equation 4.4:

$$\beta(S) = \max(\gamma_y(T) + \gamma_c(T) + \beta(to(T))) \tag{4.4}$$

   where $\beta(S_{end}) = 0$ and to represents a backward recursion.

4. The values of δ can be calculated according to equation 4.5:

$$\delta_y(T) = \gamma_c(T) + \alpha(fr(T)) + \beta(to(T)) \tag{4.5}$$

5. The uncoded bits $\tilde{y_e}/\tilde{z_e}$ are obtained in equation 4.6:

$$\tilde{y_e}/\tilde{z_e} = \max_{T|y(T)=0}(\delta_y(T)) - \max_{T|y(T)=1}(\delta_y(T)) \tag{4.6}$$

This fifth step concludes the algorithm.

4.1.2.3 CRC Check/Removal

CRC check: The goal of a CRC is to detect if an error has occurred during the decoding process. A CRC error check function uses the same polynomial (Table 3.1) as during the encoding phase and does the same operation as in the encoder. The received transport block includes a concatenated remainder calculated in sub-subsection 3.1.1.1. Therefore, the division should yield a remainder of zero, marking that no errors could be detected, with other values meaning changes occurred in data, during transmission. Figure 4.5 exemplifies a CRC check example.

Let's denote the polynomial generator by G and the received bit pattern by D' (the remainder R calculated in sub-subsection 3.1.1.1 is concatenated with data D). In this example, $G = 10011$ and $D' = 10101010100100$. When performing an "exclusive OR" operation between D' and G, it is possible to conclude the resulting remainder will be 0. Therefore, no errors occurred during the data transmission [Blo17a].

CRC removal: The generator polynomial length for NB-IoT is 25. Therefore, if no errors are detected, the last 24 bits (remainder length previously attached to the original bits) of the transport block are removed. The transmitted data is received and the decoding process is finished.

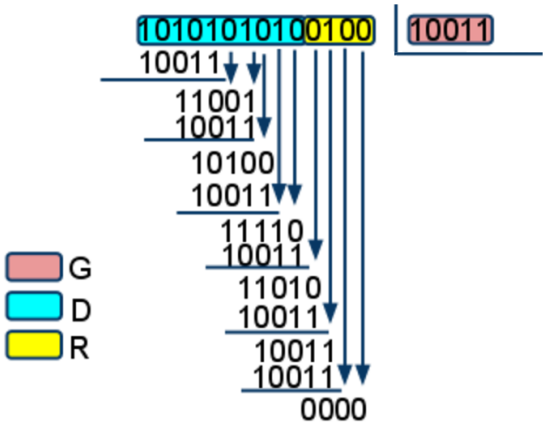


Figure 4.5: CRC check example [Blo17a].

4.2 Uplink Control Information Recovery

This section describes the baseband signal demodulation into several codewords and subsequent decoding in order to obtain a HARQ-ACK indicator on the receiver side. On the transmitter, in the SC-FDMA modulation step, all slots are repeated a certain number of times. Therefore, one codeword is obtained for each repetition. Afterwards, the several codewords are decoded and the results are combined, with the combination value being the final HARQ-ACK indicator. Figure 4.6 outlines the required steps to recover an UCI transport block. Subsections 4.2.1 and 4.2.2 describe demodulation and decoding procedures, respectively.

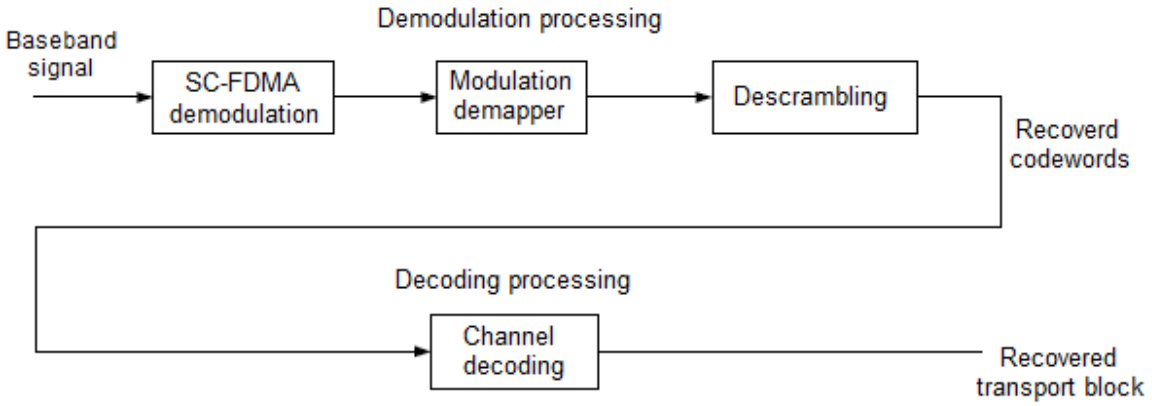


Figure 4.6: Block diagram of the NB-IoT UCI receiver [GZAM10].

4.2.1 Demodulation Processing

Demodulation processing steps are performed in exactly the same way as for the UL-SCH. Descrambling is done according to sub-subsection 4.1.1.3. Demodulation is done according to sub-subsection 4.1.1.2, always using BPSK as a demodulation scheme. SC-FDMA de-

modulation is done according to sub-subsection 4.1.1.1, always using a single subcarrier and, therefore, a single-tone transmission.

When the demodulation process is terminated, the baseband format signal is converted into several codewords.

4.2.2 Channel Decoding Processing

This subsection describes the only channel decoding step that transforms a codeword into a UCI transport block.

The control data arrives to the channel decoding unit in the form of several codewords. They are combined so the most probable bit sequence is decoded. Then, this sequence is decoded in the form of an HARQ-ACK indicator, according to Table 4.4. If HARQ-ACK equals one, the UE successfully received the data sent by the eNodeB. Otherwise, the UE failed to received the eNodeB downlink transmission.

Table 4.4: HARQ-ACK decoding.

HARQ-ACK codeword	**HARQ-ACK**
<0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>	0
<1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1>	1

4.3 Demodulation Reference Signals - Receiver

When received, the transmitted resource elements have been affected by AWGN and channel fading. Using a channel estimation, it is possible to equalize the channel effects on the received resource grid.

To facilitate the channel estimation, NB-IoT uses reference signals (pilot symbols) inserted in both time and frequency. These signals are assigned different positions within a subframe, depending on the NPUSCH format used and the selected Δf, as shown in Figures 3.14 and 3.15. The pilots provide a good estimate of the complex gains, imposed by the propagation channel, onto each grid resource element.

4.3.1 Channel Estimation

The first step in the channel estimation is to collect all the pilot symbols from their known locations within each slot. Because the value of these pilot symbols is known, a channel response estimation at these locations can be determined according to equation 4.7.

$$Y = H \times X + N \Leftrightarrow H = \frac{Y}{X} - N, \tag{4.7}$$

where Y is the received pilot symbols, X is the known/transmitted pilot symbols, H is the complex channel gain and N corresponds to AWGN noise.

After channel estimation, equalization is performed. The selected equalizer was the Zero Forcing (ZF) equalizer, due to its simplicity. When using this method, channel estimation is

calculated assuming no AWGN is present in the system. Therefore, instead of using the exact model from equation 4.7, equation 4.8 is used.

$$Y = \hat{H}_{ZF} \times X \Leftrightarrow \hat{H}_{ZF} = \frac{Y}{X}, \tag{4.8}$$

where $\hat{H}_{ZF}$ is the ZF channel estimate. Since the received pilot symbols Y, and the transmitted pilot symbols X are known, the channel can easily be estimated at each pilot position [Mat].

4.3.2 Zero Forcing Equalizer

When using the ZF equalization algorithm, the received sequence is multiplied by the multiplicative inverse of the channel frequency response.

Using equation 4.9 and the channel estimation value obtained in subsection 4.3.1, it is possible to mitigate the channel effects on the received sequence.

$$\hat{X} = \frac{Y}{\hat{H}_{ZF}}, \tag{4.9}$$

where $\hat{X}$ is the estimated transmitted signal, Y is the received signal and $\hat{H}_{ZF}$ is the channel estimate corresponding to the equalizer used.

The zero-forcing equalizer removes all ISI, and is ideal when the channel has no AWGN. However, if the channel has noise, the equalizer will amplify it.

After the equalization, the received sequence is ready to be demodulated and decoded.

4.4 Physical Random Access Channel

Using the same method as in LTE, it is possible to detect the NPRACH preamble on the receiver side. On the eNodeB, the cross-correlation between the received signal and the expected ZC sequence is calculated. If the cross-correlation value exceeds some predetermined threshold, the preamble is detected. Otherwise, the preamble is not present. There is a trade-off between preamble misdetections and false alarms. Therefore, the threshold should be set carefully [LAW16].

In this chapter, the NB-IoT receiver chain was outlined. The physical layer processing applied to the baseband signal, which leads to the transport block recovery, was described in detail. In the next chapter, an overview of the NB-IoT physical layer procedures will be presented. Required parameters, necessary for the baseband signal construction and respective detection, will be explained.

Chapter 5

NB-IoT Physical Layer Procedures

The main goal of this chapter is to explain in what circumstance the different physical channels (NPRACH, NPUSCH format 1 and 2) are used. Each one of them requires specific parameters to construct the baseband signal. Those are listed, including a description of their purpose. It's also made a brief introduction to provide some context.

5.1 Introduction

This introduction provides an overview of all the physical layer procedures mentioned in this chapter. Each one of them has a specific goal, which is summarized. The procedures are:

- **Cell search** (section 5.2) - It's the first contact between an eNodeB and a UE. Its main goal is to assign the narrowband physical layer cell identity (N_{ID}^{Ncell}) value to the UE. Since this procedure occurs on a downlink transmission, it is only briefly explained.
- **RAR procedure** (section 5.3) - It is an initialization procedure that connects the UE to the eNodeB. It happens after the cell search and is the only situation where the NPRACH is used. The subcarrier spacing (Δf) is assigned to the UE on Msg2.
- **NPUSCH format 1 UE procedure** (section 5.4) - This procedure explains in what circumstance the UE transmits data on the NPUSCH format 1. Generally, when receiving a Narrowband Physical Downlink Control Channel (NPDCCH) format N0 from the eNodeB, the UE receives necessary parameters for the construction of the baseband signal.
- **NPUSCH format 2 UE procedure** (section 5.5) - Before a Narrowband Physical Downlink Shared Channel (NPDSCH) transmission, the eNodeB supplies the UE with parameters required for the decoding/demodulation of that transmission on the NPDCCH format N1. Afterwards, the UE sends a positive or negative acknowledgment using the NPUSCH format 2.

5.2 Cell Search

This section briefly describes the cell search procedure's goal. Its main objective is the N_{ID}^{Ncell} detection by the UE. N_{ID}^{Ncell} has 504 possible values, ranging from 0 to 503.

In NB-IoT, the Narrowband Primary Synchronization Signal (NPSS) and Narrowband Secondary Synchronization Signal (NSSS) are transmitted in the downlink to facilitate the cell search. Since the N_{ID}^{Ncell} detection occurs on a downlink transmission, it is only provided a brief description of this procedure.

5.3 RAR Procedure

This section describes the steps required to perform the RAR procedure (Figure 5.1).

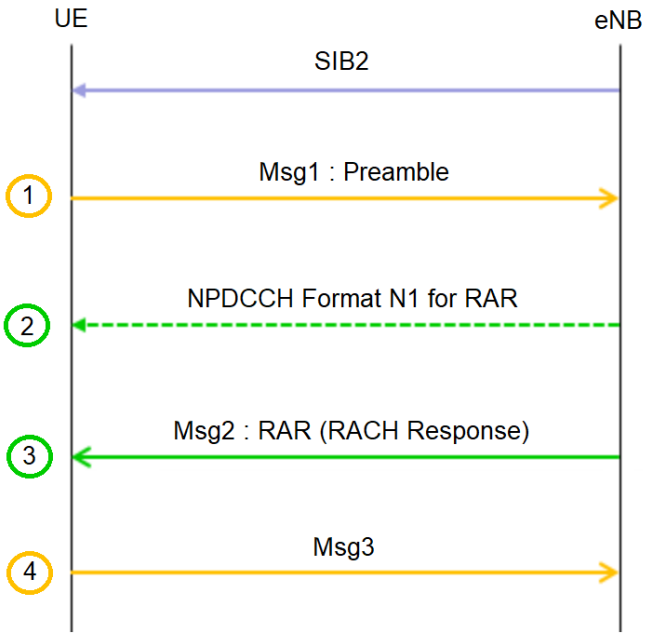


Figure 5.1: Overall RAR procedure. SIB2 transmission is required before the RAR procedure itself.

Prior to initiation of the RAR procedure, the UE should receive a System Information Block (SIB)2 (purple arrow in Figure 5.1), sent with the necessary values to create a NPRACH preamble. The following information is necessary [3GP16]:

- The frequency location of the first subcarrier allocated to NPRACH ($N_{scoffset}^{NPRACH}$).
- The number of subcarriers allocated to NPRACH (N_{sc}^{NPRACH}).
- The number of NPRACH repetitions per attempt (N_{rep}^{NPRACH}).

It is of extreme importance to receive this information block, since this message sets several constants used by the UE on the RAR procedure. Each step of the actual procedure is enumerated in Figure 5.1 inside a circle.

1. The first step consists in establishing communication between the UE and the eNodeB. To do this, a preamble, generated as described in section 3.4, is sent on the NPRACH physical channel. The preamble is repeated N_{rep}^{NPRACH} times as indicated in SIB2.

2. The second step consists on the detection of a NPDCCH, where posterior demodulation/decoding leads to the Downlink Control Information (DCI) format 1 values.

   A format N1 message is composed of 23 bits (Figure 5.2). Starting with the Most Significant bit (MSb), the following information is received [3GP17a]:

   - 1 bit corresponding to the DCI format flag:
     - '0' for format N0 (section 5.4).
     - '1' for format N1.
   - 1 bit corresponding to the NPDCCH order indicator.
     - '1' means that this information is used for NPRACH scheduling.
     - '0' denotes the information is used for NPDSCH scheduling (section 5.5).
   - 2 bits corresponding to the number of repetitions.
   - 6 bits corresponding to the subcarrier indication.
   - 13 remaining bits that are set to '1'.

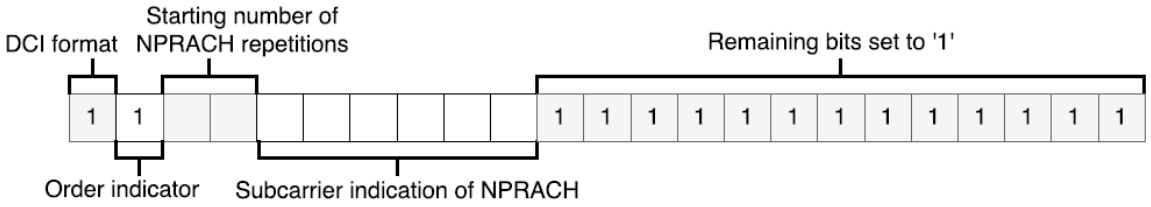


Figure 5.2: Contents of DCI format N1 when is used for scheduling NPRACH [DT17].

3. The third step is the decoding/demodulation of Msg2 (downlink transmission), using the information received on step 2. The baseband signal is parsed, indicating the Nr-bit uplink grant to the physical layer, which contains 15 bits.

   Its content, starting with the MSb is as follows [3GP17b]:

   - 1 bit corresponding to the uplink subcarrier spacing (Δf):
     - '0' for 3.75 kHz.
     - '1' for 15 kHz.

     The Δf value is only indicated in Msg2 (RACH Response). In all other transmissions, this is the value to be used.
   - 6 bits corresponding to the subcarrier indication field (I_{sc}).
   - 2 bits corresponding to the scheduling delay indication field (I_{Delay}).
   - 3 bits corresponding to the repetition number indication field (I_{Rep}).
   - 3 bits corresponding to the modulation and coding scheme indication field (I_{MCS}), which indicates the TBS, the modulation scheme, and the total number of RUs for Msg3.

4. In the last step, if a NPDCCH was detected and its corresponding transport block consists on a response to the transmitted preamble sequence, the UE transmits an UL-SCH transport block according to section 3.1, where the required parameters are given on Msg2. It's important to note the redundancy version indication field (rv_{dci}) for the transmission of a Msg3 is 0.

   If the corresponding Downlink Shared Channel (DL-SCH) transport block does not contain a response to the transmitted preamble sequence, the UE transmits the preamble sequence again, going back to the first step.

5.4 NPUSCH Format 1 UE procedure

This section describes the steps required to perform the NPUSCH format 1 UE procedure. In Figure 5.3, each step is enumerated inside a circle.

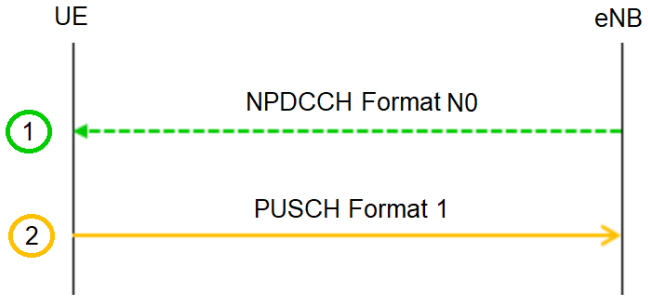


Figure 5.3: Overall NPUSCH format 1 UE procedure.

1. The first step consists on the detection of a NPDCCH by the UE, which after parsing supplies the DCI format N0, that includes relevant parameters to send a transport block on the NPUSCH. Besides the DCI format N0 values, this transmisison gives the UE the Radio Network Temporary Identifier (RNTI). When the NPDCCH is codified, in the CRC addition step, the RNTI parameter is used to scramble the obtained CRC bits.

   DCI format N0 is composed of 23 bits ordered according to Figure 5.4. Starting with the MSb, the following information is received [3GP17a]:

   - 1 bit corresponding to the DCI format flag:
     - '0' for format N0.
     - '1' for format N1 (section 5.5).
   - 6 bits corresponding to the subcarrier indication field (I_{sc}).
   - 3 bits corresponding to the resource assignment indication field (I_{RU}).
   - 2 bits corresponding to the scheduling delay indication field (I_{Delay}).
   - 4 bits corresponding to the modulation and coding scheme indication field (I_{MCS}). This value, combined with I_{RU} allows the TBS calculation.
   - 1 bit corresponding to the redundancy version indication field (rv_{dci}).

- 3 bits corresponding to the repetition number indication field (I_{Rep}).
- 1 bit corresponding to the new data flag.
- 2 bits corresponding to the DCI subframe repetition number.

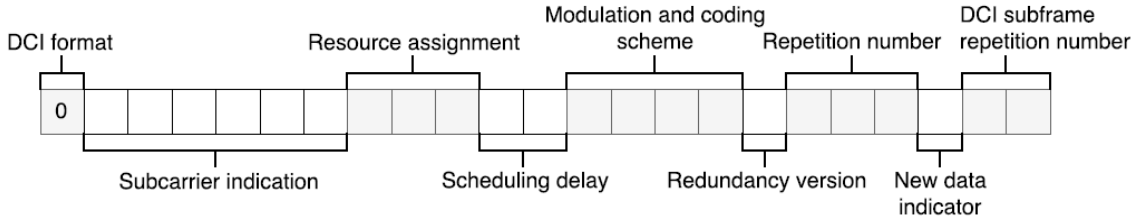


Figure 5.4: Contents of DCI format N0 [DT17].

2. The second step consists in sending the corresponding NPUSCH format 1 in $N_{Rep} \times N_{RU} \times N_{slots}^{UL}$ consecutive slots. The value of N_{slots}^{UL} is the number of slots in the selected resource unit given in Table 2.2. The N_{RU} is determined by the resource assignment field in the corresponding DCI, according to Table 5.1.

   The number of subcarriers allocated to NPUSCH (n_{sc}) is given by the I_{sc} present in the DCI format N0, determined according to Table 5.2.

Table 5.1: N_{RU} obtained according to I_{RU} value.

I_{RU}	N_{RU}
0	1
1	2
2	3
3	4
4	5
5	6
6	8
7	10

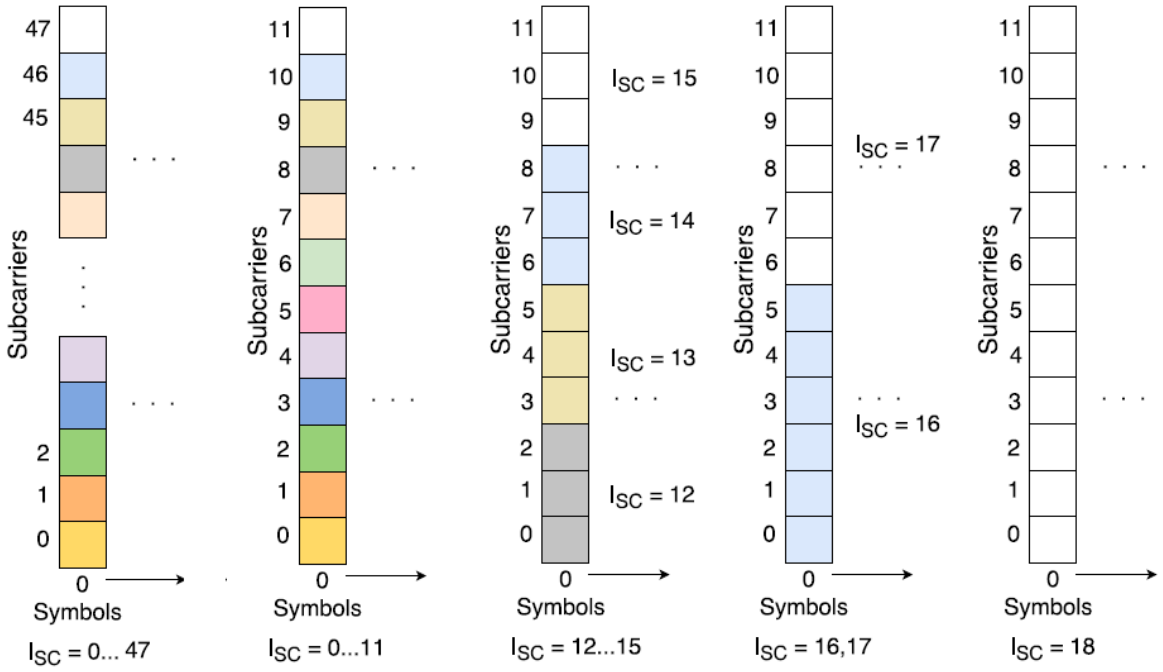
Table 5.2: Values of allocated subcarriers n_{sc} depending on the subcarrier indicator I_{sc}.

Subcarrier spacing (Δf)	I_{sc}	n_{sc}
3.75kHz	0-47	I_{sc}
	48-63	Reserved
15kHz	0-11	I_{sc}
	12-15	$3(I_{sc} - 12) + 0, 1, 2$
	16-17	$6(I_{sc} - 16) + 0, 1, 2, 3, 4, 5$
	18	{0,1,2,3,4,5,6,7,8,9,10,11}
	19-63	Reserved

When Δf is 3.75kHz, a range of values from 0-47 assigns the subcarrier to be used. An example is shown on Figure 5.5a. If Δf is 15kHz, a range of values from 0-18 assigns the subcarriers to be used. If I_{sc} is bigger then 11, several subcarriers are utilized. 12, 13, 14 and 15 indicates three subcarriers are used, corresponding to one of four quarters of the resource block. 16 and 17 indicate either the top or bottom half of the resource block is selected. 18 indicates all the subcarriers are chosen. This arrangement is depicted in Figure 5.5b.

The modulation order (Q_m) and the transport block size indication field (I_{TBS}) is determined using the the I_{MCS}, according to Table 5.3.

The total number of repetitions (N_{Rep}) is determined by the I_{Rep} present in the DCI format N0, according to Table 5.4. One transport block can be repeated several times.



(a) Allocated subcarriers as indicated by n_{sc} when $\Delta f = 3.75$kHz.

(b) Allocated subcarriers as indicated by n_{sc} when $\Delta f = 15$kHz.

Figure 5.5: Allocated subcarriers as indicated by n_{sc} [DT17].

Table 5.3: Q_m and I_{TBS} obtained according to the I_{MCS} value.

Number of subcarrier (N_{sc}^{RU})	I_{MCS}	Q_m	I_{TBS}
3/6/12 subcarriers	0-12	2	I_{MCS}
1 subcarrier	0	1	0
	1	1	2
	2	2	1
	3	2	3
	4	2	4
	5	2	5
	6	2	6
	7	2	7
	8	2	8
	9	2	9
	10	2	10

Table 5.4: N_{Rep} obtained according to the I_{Rep} value.

I_{Rep}	N_{Rep}
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

The arrangement of the repetitions depends on number of consecutive subcarriers in a resource unit (N_{sc}^{RU}), the subcarrier spacing (Δf) and the total number of repetitions (N_{Rep}). This is exemplified in Figure 5.6.

The description of Figure 5.6 goes as follows:

(a) Let's consider Δf is 15kHz and a transport block is transmitted on two RUs, named T and W.

(b) Each RU has eight slots and, consequently, three subcarriers - Table 2.2. T_1 corresponds to the first slot of the first RU and W_1 corresponds to the first slot of the second RU, and so forth.

(c) Let's assume a total number of eight repetitions is applied ($N_{Rep} = 8$) First, slots T_1 and T_2 are transmitted. This pair is repeated three more times. Therefore, four transmissions of these slots occur. This procedure is continued until the slots W_7 and W_8 are pairwise transmitted four times. Finally, the transmission sequence is repeated once again, reaching the eight repetitions.

When Δf is 15kHz, the first repetition of two slots is always done - number of grouped slots (N_{slots}) = 2. When Δf is 3.75kHz, it is done for every slot separately - number of grouped slots (N_{slots}) = 1. If the RU has more than one subcarrier ($N_{sc}^{RU} > 1$), the number of repetitions of grouped slots ($M_{identical}^{NPUSCH}$) is half the number of N_{Rep}, with an upper limit of four. Otherwise, if the RU has only one subcarrier, this value is one.

For instance, in Figure 5.6, if there would be 64 repetitions, the $M_{identical}^{NPUSCH}$ number would be the same. However, the total sequence would be repeated 15 additional times - scheduled number of repetitions of a NPUSCH transmission (M_{Rep}^{NPUSCH}) = 16 [Roh16].

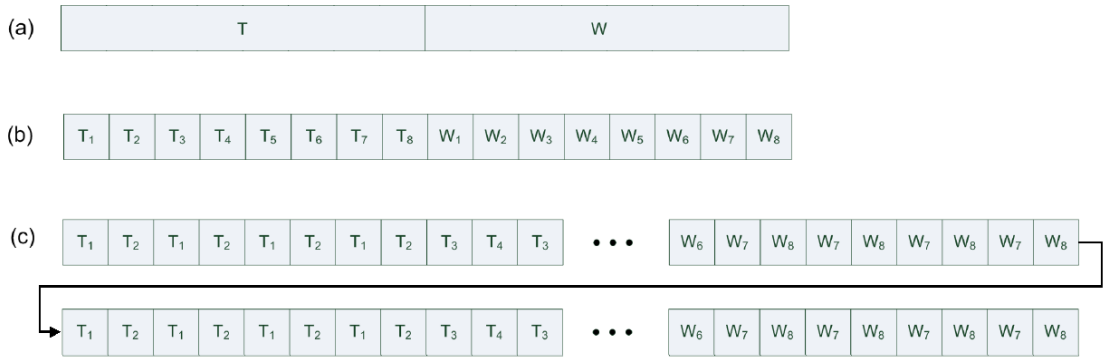


Figure 5.6: Example of an arrangement for NPUSCH transmission with 8 repetitions. For the case of no repetitions, the slot sequence shown in (b) would be transmitted [Roh16].

The I_{TBS}, used in combination with the I_{RU}, determines the TBS according to Table 5.5 [3GP17b].

Table 5.5: TBS obtained according to I_{TBS} and I_{RU} values.

I_{TBS}	I_{RU}							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	16	32	56	88	120	152	208	256
1	24	56	88	144	176	208	256	344
2	32	72	144	176	208	256	328	424
3	40	104	176	208	256	328	440	568
4	56	120	208	256	328	408	552	680
5	72	144	224	328	424	504	680	872
6	88	176	256	392	504	600	808	1000
7	104	224	328	472	584	712	1000	-
8	120	256	392	536	680	808	-	-
9	136	296	456	616	776	936	-	-
10	144	328	504	680	872	1000	-	-
11	176	376	584	776	1000	-	-	-
12	208	440	680	1000	-	-	-	-

rv_{idx} is determined by the resource assignment field in the corresponding DCI according to equation 5.1.

$$rv_{idx} = 2 \times \mathrm{mod}(rv_{dci}, 2) \tag{5.1}$$

Coding and modulation of the NPUSCH format 1 physical channel is done according to section 3.1.

5.5 NPUSCH Format 2 UE procedure

This section describes the steps required to perform the NPUSCH format 2 UE procedure. In Figure 5.7, each step is enumerated inside a circle.

1. The first step consists on the detection, by the UE, of a NPDCCH with a DCI format N1 that includes all relevant parameters to receive, demodulate and decode a NPDSCH transport block. Besides the DCI format N0 values, this transmission gives the UE the RNTI. When the NPDCCH is codified, in the CRC addition step, the RNTI parameter is used to scramble the obtained CRC bits.

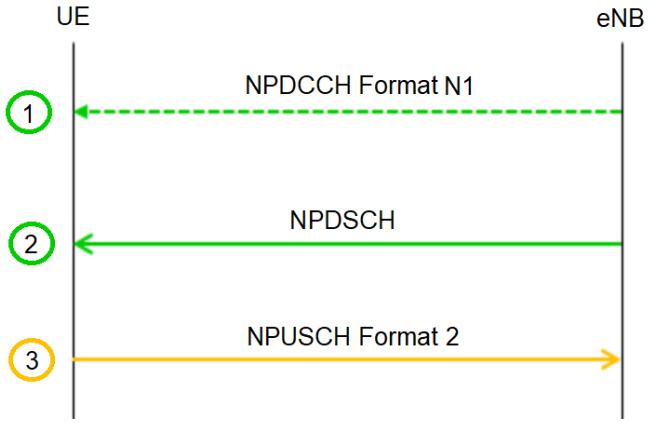


Figure 5.7: Overall NPUSCH format 2 UE procedure.

DCI format N1 is composed of 23 bits, represented in Figure 5.8. Starting with the MSb, the following information is received [3GP17a]:

- 1 bit DCI format flag:
  - '0' for format N0 (section 5.4).
  - '1' for format N1.
- 1 bit NPDCCH order indicator:
  - '1' means that this information is used for NPRACH scheduling (section 5.3).
  - '0' denotes the information is used for NPDSCH scheduling.
- 3 bits corresponding to the scheduling delay indication field (I_{Delay}).
- 3 bits corresponding to the resource assignment indicator (I_{SF}), with the value of scheduled downlink frames.
- 4 bits corresponding to the modulation and coding scheme indication field (I_{MCS}).
- 4 bits corresponding to the repetition number indication field (I_{Rep}).
- 1 bit corresponding to the new data indicator flag.
- 4 bits corresponding to the Acknowledgement/Negative-Acknowledgement (ACK-/NACK) resource field.
- 2 bits corresponding to the Downlink Control Information (DCI) subframe repetition number.

All values are used on the reception and subsequent decoding of the NPDSCH, except for the ACK/NACK resource field. Instead, it's a parameter necessary to send a positive or negative acknowledgment on NPUSCH format 2.

2. Detection of a NPDSCH transmission and parsing of the received signal, using the information provided on the DCI format N1.

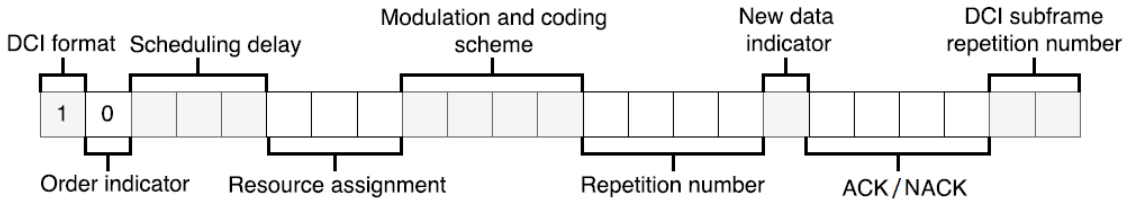


Figure 5.8: Contents of DCI format N1 when used for scheduling NPDSCH [DT17].

3. Upon detection of a NPDSCH transmission by the UE, an acknowledgment should be provided using NPUSCH format 2 in $N_{Rep}^{AN} \times N_{slots}^{UL}$ consecutive slots, with ACK/NACK number of repetitions (N_{Rep}^{AN}) being a higher layer parameter. The N_{slots}^{UL} is always four and the number of consecutive subcarriers in a resource unit (N_{sc}^{RU}) is always one - Table 2.3. The total number of resource units (N_{RU}) is, also, always one.

   The allocated subcarrier is determined by the ACK/NACK resource field according to Table 5.6, when Δf is 3.75kHz, and Table 5.7, when Δf is 15kHz [3GP17b].

Table 5.6: NPUSCH format 2 allocated subcarrier when $\Delta f = 3.75$kHz.

ACK/NACK resource field	**ACK/NACK subcarrier**
0	38
1	39
2	40
3	41
4	42
5	43
6	44
7	45
8	38
9	39
10	40
11	41
12	42
13	43
14	44
15	45

Table 5.7: NPUSCH format 2 allocated subcarrier when $\Delta f = 15$kHz.

ACK/NACK resource field	**ACK/NACK subcarrier**
0	0
1	1
2	2
3	3
4	0
5	1
6	2
7	3
8	0
9	1
10	2
11	3
12	0
13	1
14	2
15	3

Coding and modulation of the NPUSCH format 2 physical channel is done according to section 3.2.

Throughout the first five chapters, the NB-IoT physical layer was described. First, general concepts were introduced. Then, the transmitter and receiver chains for all physical channels were explained. To connect the introduced concepts, this fifth chapter was written, which explains in what circumstances a physical channel is used and the required parameters to generate its corresponding baseband channel.

On the next chapter, it is introduced how the MATLAB simulation of these theoretical concepts was implemented. Furthermore, it is detailed how a co-simulation using USRPs as RF front-ends was developed, explaining its utility.

Chapter 6

NB-IoT MATLAB Modeling and USRP Co-Simulation

The main goal of this dissertation consisted on the implementation of a behavioral modeling of the NB-IoT uplink physical layer. This chapter starts by giving a brief overview of what was implemented, explaining why MATLAB was chosen as a simulation environment. Afterwards, it describes, with the aid of a block diagram, the main functions used to convert data into a baseband signal, simulate the channel and recover the original data. Finally, it clarifies how two USRPs, used as RF front-ends in a co-simulation environment, can transmit/receive said baseband signal.

6.1 Introduction

This section briefly describes what was implemented during this dissertation's practical work. There are two main parts. First, a MATLAB model of the NB-IoT uplink physical layer was designed. Both transmitter and receiver chains, including all coding/decoding and modulation/demodulation steps were implemented.

Secondly, a software/hardware co-simulation using two USRPs as RF front-ends, was tested. One USRP is utilized to transmit the baseband signal generated in the MATLAB simulation. The other captures the transmitted signal to be posteriorly decoded/demodulated, using the MATLAB implementation. This process happens in real-time, which provides a degree of comparison between the simulation environment, where no time constraints need to be met.

NB-IoT can have SISO or SIMO antenna mapping on the uplink. Although that is taken into consideration throughout the simulation and co-simulation implementations, up to this point, only the SISO functionalities are completely functional.

As previously mentioned, the simulator is developed using MATLAB. This choice was influenced by three key factors. First, mathematical operations in other languages require function calls, instead of natural operators. Therefore, it takes longer to write C/C++ code equivalent to the MATLAB one. Secondly, MATLAB can be good for modeling due to its visualization capabilities. This will accelerate a posterior phase, where the NB-IoT protocol is implemented in hardware. Lastly, MATLAB Coder^{TM} generates readable and portable C/C++ code from MATLAB, which makes a posterior C/C++ implementation much simpler. Toolboxes are not used on the simulation, since it helps with the porting to another

programming language.

More information about the MATLAB implementation is provided on section 6.2. Details regarding the hardware co-simulation can be found on section 6.3.

6.2 MATLAB Implementation

This section provides a description of the MATLAB simulation. Three different simulations were implemented, one for each existent NB-IoT physical channel. Each channel has a different purpose, so each simulation uses distinct functions. Therefore, their respective block diagram also varies. Figure 6.1a represents the NPUSCH format 1 and 2 simulation block diagram. The difference between formats is in the function name that ends in `F1` and `F2`, accordingly, and in the input and output parameters. The format selection is a user input variable and, then, the functions represented on the block diagram are called in order, accordingly to the previously selected format. The gray dotted squares of Figure 6.1a are only used on the co-simulation implementation, explained on section 6.3. Both simulations are described in further detail in subsections 6.2.2 and 6.2.3.

Figure 6.1b represents the block diagram of the NPRACH implementation. This is an entirely different simulation and is not integrated with the main one. This will be explained in further detail in subsection 6.2.4.

6.2.1 Notation

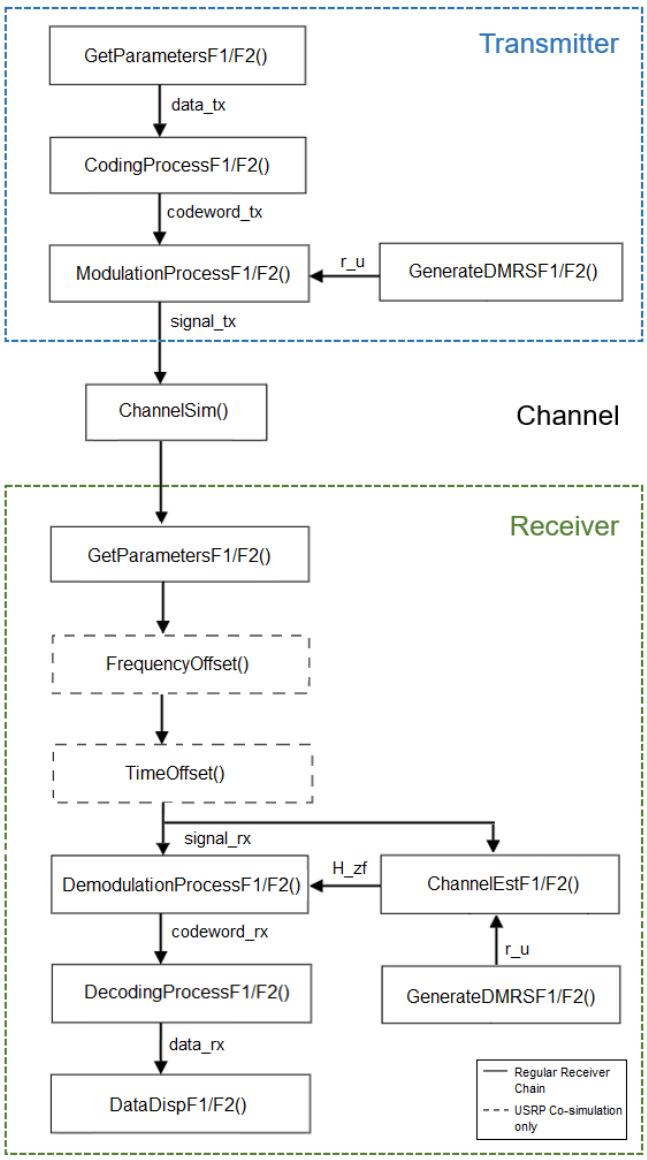
On the MATLAB implementation, parameters are represented according to their name in the official release specs. If the parameter has a superscript, it is represented in upper-case letters. If it has a subscript, it is written in lower-case letters. The order is always superscript and then subscript, if the variable has both. For example, $N_{symbols}^{UL}$ would be `NULsymbols`, n_{sc} would be `Nsc`, and N_{RU} would be `Nru`. Generally, upper-case 'N' represents number, 'UL' stands for uplink and 'RU' means resource unit. Parameters that are not mentioned in the release and are only part of the MATLAB simulation always have an underscore. Those followed by a 'tx', always have their equivalent on the receiver chain, being followed by 'rx'. For example, the baseband signal `signal_tx` has an equivalent called `signal_rx`.

6.2.2 NPUSCH Format 1 simulation

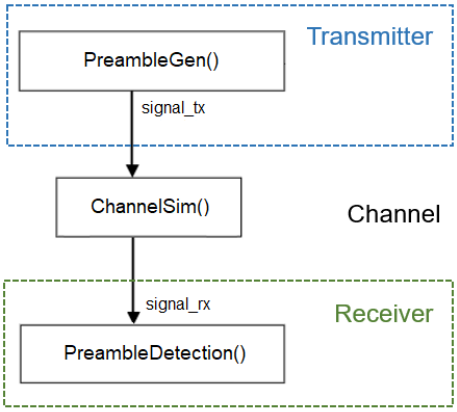
This subsection describes the NPUSCH format 1 simulation, being divided in transmitter, channel and receiver. Each one of them is explained in depth on sub-subsections 6.2.2.1, 6.2.2.2 and 6.2.2.3.

6.2.2.1 Transmitter Implementation

The **transmitter** is represented in Figure 6.1a by the blue dotted lines. All the coding and modulation procedures described in section 3.1 were implemented. DCI format N0 parameters are generated as user input. The used subcarrier spacing (`deltaf`) is also decided by the user. If the user leaves them empty or chooses an impossible combination of values, they are automatically assigned, when needed. Using the DCI values, it is possible to calculate a set of required parameters to obtain the baseband signal, according to section 5.4. This is



(a) Block diagram of the NPUSCH format 1 and 2 physical channel MATLAB simulation.



(b) Block diagram of the NPRACH physical channel MATLAB simulation.

Figure 6.1: Block diagrams of the MATLAB implemented simulations.

done using the function GetParametersF1(). Input and output parameters are represented in Table 6.1.

Table 6.1: Function GetParametersF1() description.

Function	GetParametersF1()
Input	deltaf, Irep, Imcs, Iru, Isc, RVdci, RNTI
Output	data_tx, Qm, RVidx, Nsc, MNPUSCHrep, MNPUSCHidentical, Nru, Nslots, NRUsc, NULsc, TBS, RNTI, NULslots, NULsymbols, E, Ns, Nsf, Nf, NNCELLid

Outputs Qm, RVidx, Nsc, MNPUSCHrep, MNPUSCHidentical, Nru, Nslots, NRUsc, RNTI and TBS are calculated using the user input values, according to section 5.4. NULsc is dependent on the deltaf value, chosen by the user. NULslots is determined based on Table 2.2 and NULsymbols is always 7. data_tx with length TBS is the randomly generated information to be sent. Ns, Nsf and Nf counters, corresponding to the number of slots, subframes and frames, respectively, are initialized with value zero. NNCELLid has a range of values between 0 and 503. Since its calculation is a downlink procedure, and for code simplification, this value is always 1 (can easily be changed by the user).

Considering the modulation order (Qm), the number of resource units available (Nru) and how many subcarriers are used in each (NRUsc), the maximum number of bits that is possible to send in the available space is calculated. Sometimes, the puncturing of several bits is required to fit in that space. Variable E is the maximum number of bits possible to send if the turbo encoder output exceeds that size. If enough space is available and no puncturing is necessary, E corresponds to the turbo encoder output length.

Function CodingProcessF1() performs the steps described in subsection 3.1.1 - CRC addition, turbo encoding and rate matching, with Table 6.2 showing its inputs and outputs. In this function, the transport block data_tx is coded into the codeword codeword_tx. Rate matching requires E and RVidx parameters.

Table 6.2: Function CodingProcessF1() description.

Function	CodingProcessF1()
Input	data_tx, RVidx, E
Output	codeword_tx

Function GenerateDMRSF1() generates the DMRS in order to facilitate the transport block recovery on the receiver side. More information about the generation of DMRS can be found on section 3.3.

Table 6.3: Function Generate_DMRSF1() description.

Function	Generate_DMRSF1()
Input	Nru, NRUsc, NULslots, NNCELLid, group_hopping
Output	ru

According to Table 6.3, the function receives as input NRUsc, NNCELLid, Nru and NULslots, since the formulas used to calculate this signals depend on the mentioned variables.

`group_hopping` is a user defined parameter which activates or deactivates the group hopping on the DMRS generation.

Function `ModulationProcess()` performs modulation, scrambling and SC-FDMA signal generation (including resource mapping) according to section 3.1.2. Furthermore, it repeats each slot according to `MNPUSCHrep` and `MNPUSCHidentical` values and subsection 5.4.

Table 6.4: Function `ModulationProcess()` description.

Function	`ModulationProcess()`
Input	`codeword_tx, ru, Qm, Nsc, MNPUSCHrep, MNPUSCHidentical, RNTI Nslots, Nru, NRUsc, NULsc, NULslots, NULsymbols, Ns, Nf, NNCELLid`
Output	`signal_tx`

Table 6.4 shows the input and output parameters of the function. `RNTI`, `NNCELLid`, `Nf` and `Ns` are used on the scrambling initialization formula. `Qm` is necessary to decide which modulation scheme to use. `Nsc`, `Nru`, `NRUsc`, `NULsc`, `NULslots` and `NULsymbols` are utilized on the resource mapping of the data and the pilot symbols (`ru`). `MNPUSCHrep`, `MNPUSCHidentical` and `Nslots` are used on the repetitions of each slot, as described in section 5.4. In the end, the codeword, `codeword_rx`, is modulated into the generated signal, `signal_tx`.

6.2.2.2 Channel Implementation

In this sub-subsection, it's introduced some necessary knowledge about the used channel models. A suitable model should be chosen, so the simulation is as close to reality as possible.

AWGN channel model: The first channel model is used when no fading or multipath propagation is present, only adding AWGN to the signal. The received signal is represented as $y = x + n$, where n represents the noise contributed by AWGN, which is Gaussian distributed with zero mean.

The MATLAB function `'randn'` generates normally distributed random numbers with a mean of 0 and a variance of 1. The output is scaled so the result has the desired variance (which depends on E_b/N_0).

Fading channel model: In a wireless communications system, the transmitted signal always suffers from multipath fading. To represent the multipath component, three different delay profiles are suggested by 3GPP. These are Extended Pedestrian A (EPA), Extended Vehicular A (EVA) and Extended Typical Urban (ETU), with each tap delay and respective power represented on Tables 6.5, 6.6 and 6.7. Furthermore, a maximum Doppler frequency is specified for each one of them, corresponding to 5Hz, 70Hz and 300Hz, respectively. It can be noticed that the EPA model has 7 multipath components, while the EVA and ETU models have 9 multipath components each. MATLAB's built-in function `'rayleighchan'` is used to implement the three channel models. Therefore, the received signal is represented as $y = Hx$, where H is the channel response.

Fading channel with AWGN model: In this model, AWGN is also added to the three fading channel possibilities. Therefore, the received signal is represented as $y = Hx + n$, where H is the channel response and n is the AWGN.

Table 6.5: EPA Delay Profile.

Excess tap delay [ns]	**Relative Power [dB]**
0	0.0
30	-1.0
70	-2.0
90	-3.0
110	-8.0
190	-17.2
410	-20.8

Table 6.6: EVA Delay Profile

Excess tap delay [ns]	**Relative Power [dB]**
0	0.0
30	-1.5
150	-1.4
310	-3.6
370	-0.6
710	-9.1
1090	-7.0
1730	-12.0
2510	-16.9

Table 6.7: ETU Delay Profile.

Excess tap delay [ns]	**Relative Power [dB]**
0	-1.0
50	-1.0
120	-1.0
200	0.0
230	0.0
500	0.0
1600	-3.0
2300	-5.0
5000	-7.0

The **Channel** is represented in Figure 6.1a, between the transmitter and the receiver sections. The input and output of the function is represented in Table 6.8. The channel model to use is decided based on `mode` value. If it has the value 1, only AWGN is added. If it has the values 2, 3 or 4, the EPA, EVA and ETU fading models are used, respectively. If the values are 5, 6 or 7, EPA, EVA and ETU fading models are used, but AWGN is also added. Finally, there is an extra mode, where no interference whatsoever is added and the

transmitted and received signals are the same (mode= 8).

Additionally as user input is the EbN0, used to calculate the variance, when necessary. Table 6.8 represents the ChannelSim() function inputs and output.

Table 6.8: Function ChannelSim() description.

Function	ChannelSim()
Input	signal_tx, EbN0, mode
Output	signal_rx

6.2.2.3 Receiver Implementation

The **Receiver** is represented in Figure 6.1a by the green dotted lines. The functions inside the gray dotted squares are bypassed, since they are only necessary for the USRP co-simulation testing - section 6.3.

The first steps on the receiver implementation are the general parameters and the DMRS (pilot symbols) generation. These functions are equal on both the transmitter and the receiver sides.

GetParametersF1() corresponds to the first function (Table 6.1) and the reason why it is called on both sides is so the simulation works when two different computers are used. Obviously, the input values have to be the same on the transmitter and the receiver. This implementation is accurate, since the eNodeB always informs the UE on what values to use (discussed in section 5.4). When calling this function on the receiver side, the output data_tx is empty.

The second function is called GenerateDMRSF1() and its inputs and output are represented on Table 6.3. Afterwards, it is possible to estimate the channel based on the received pilot symbols that are compared with the calculated ones. To do this, function ChannelEstF1() is called and its inputs and outputs are represented in Table 6.9.

Table 6.9: Function ChannelEstF1() description.

Function	ChannelEstF1()
Input	data_rx, ru, NULsc, Nsc, NRUsc, NULsymbols, NULslots, Nslots, MNPUSCHidetical, MNPUSCHrep
Output	H_zf

First, a reference grid is generated using the know pilot values (ru - output of GenerateDMRSF1()), with the unknown values being zero. To generate the grid, NULsc, Nsc, NRUsc, NULsymbols, NULslots, Nslots, MNPUSCHidentical and MNPUSCHrep are necessary. Then, data_rx is also rearranged in grid format. Finally, by comparing both grids, H_zf is calculated, according to subsection 4.3.1.

Function DemodulationProcessF1() performs SC-FDMA signal recovery (including resource de-mapping), demodulation and descrambling according to section 4.1.1. Furthermore, it takes into account the total number of repetitions (Nrep). This means it separates the signal in its several repetitions, with each one being separately demodulated. Thus, one codeword per repetition is obtained.

Table 6.10: Function `DemodulationProcessF1()` description.

Function	`DemodulationProcessF1()`
Input	`signal_rx, NULslots, NULsymbols, Nslots, NULsc, NRUsc, NNCELLid, MNPUSCHidentical, MNPUSCHrep, Nsc, Nru, Qm, RNTI, H_zf, E, Ns, Nsf, Nf`
Output	`codeword_rx, Ns, Nsf, Nf`

Table 6.10 shows the input and output parameters of the function. `RNTI`, `NNCELLid`, `Nf` and `Ns` are used to perform descrambling. `Qm` is necessary to decide which demodulation scheme to use. `MNPUSCHidentical`, `MNPUSCHRep` and `Nslots` are used to know the location of each repeated stream and separate them. `Nsc`, `Nru`, `NULsymbols`, `NULslots`, `NULsc`, `NRUsc` and `E` are important for the resource demapping. `H_zf` is utilized in the equalization step. In the end, each recovered codeword is stored in the matrix, `codeword_rx`. `Ns`, `Nsf` and `Nf` values are updated.

Function `DecodingProcessF1()` performs the steps described in section 4.1.2 - CRC removal, turbo decoding and rate dematching, with Table 6.11 showing its inputs and outputs. Rate dematching requires `TBS` and `RVidx` parameters. In this function, each codeword on matrix `codeword_rx` is decoded, becoming a transport block. Matrix `data_err` represents all the recovered transport blocks, even if the CRC check part showed errors were present. Matrix `data_noerr` shows only the transport blocks where no errors were detected by the CRC check step. `iteration_index` displays how many iterations were necessary to decode each repetition on the turbo decoder step. `errors` signals which bit streams didn't pass the CRC check step. All inputs and outputs of the function are displayed on Table 6.11.

Table 6.11: Function `DecodingProcessF1()` description.

Function	`DecodingProcessF1()`
Input	`codeword_rx, RVidx, TBS`
Output	`data_rx, data_noerr, data_err, iteration_index, errors`

The recovered transport block `data_rx` (mathematical mode of the several repetitions on matrix `data_noerr`), matrices `data_noerr` and `data_err` with the corresponding iteration indexes (`iteration_index`) and which decoded transport blocks were wrong (`errors`) are displayed when calling the function `DataDispF1()`. The number of slots (`Ns`), subframes (`Nsf`) and frames (`Nf`) utilized is also shown. Inputs of the function are shown on Table 6.12.

Table 6.12: Function `DataDispF1()` description.

Function	`DataDispF1()`
Input	`data_rx, data_noerr, data_err, iteration_index errors, Ns, Nsf, Nf`
Output	-

6.2.3 NPUSCH Format 2 Simulation

The simulation is divided in transmitter, channel and receiver. Each one of them is explained in depth on sub-subsections 6.2.3.1, 6.2.3.2 and 6.2.3.3.

6.2.3.1 Transmitter Implementation

The **transmitter** is represented in Figure 6.1a by the blue dotted lines. Required parameters are generated as user input. If the user leaves them empty or chooses an impossible combination of values, they are automatically assigned, when needed. After, it's possible to obtain a set of required parameters to design the transport block. This is done using the function `GetParametersF2()`. Input and output parameters are represented in Table 6.13.

Table 6.13: Function `GetParametersF2()` description.

Function	`GetParametersF2()`
Input	`deltaf, ACKNACK, RNTI, NANRep`
Output	`Qm, Nsc, NULslots, NULsymbols, NRUsc, Nru, NULsc NNCELLid, HARQACK, NANrep, RNTI, Ns, Nsf, Nf`

Outputs `RNTI` and `Nsc` are calculated using the user input DCI values according to section 5.5. `NULsymbols` is always 7 and `NULslots` is determined based on Table 2.3. `NULsc` is dependent on the `deltaf` value, which is chosen by the user. `Ns`, `Nsf` and `Nf` counters, corresponding to the number of slots, subframes and frames, respectively, are initialized with value 0. Both `Nru` and `NRUsc` are always 1. `HARQACK` is an indicator and has the value 1 if the downlink transport block sent is received with success, or 0, otherwise. This parameter is randomly selected. `NANrep` is a higher layer parameter between 0 and 7, chosen as a user input. `NNCELLid` has a range of values between 0 and 503. Since its calculation is a downlink procedure and for simplification, this value is always 1 (can easily be changed by the user).

Function `CodingProceduresF2()` performs the coding step described in subsection 3.2.1, with the `HARQACK` indicator being coded into codeword, `codeword_rx`. Table 6.14 shows the function input and output.

Table 6.14: Function `CodingProceduresF2()` description.

Function	`CodingProceduresF2()`
Input	`HARQACK`
Output	`codeword_rx`

Function `GenerateDMRSF2()` generates the DMRS in order to facilitate the transport block recovery on the receiver side. More information about the generation of DMRS can be found on section 3.3.

Table 6.15: Function `Generate_DMRSF2()` description.

Function	`Generate_DMRSF2()`
Input	`NRUsc, NNCELLid, Nru, NULslots, group_hopping`
Output	`ru`

According to Table 6.15, the function receives as inputs `NRUsc`, `NNCELLid`, `Nru` and `NULslots`, since the formulas used to calculate these signals depend on the mentioned variables. `group_hopping` is always empty, since for NPUSCH format 2 group hopping is always deactivated.

Function `ModulationProcessF2()` performs modulation, scrambling and SC-FDMA signal generation (including resource mapping) according to section 3.2.2. Furthermore, it repeats the only resource unit available `NANrep` times.

Table 6.16: Function `ModulationProcessF2()` description.

Function	`ModulationProcessF2()`
Input	`codeword_tx, Nsc, NANrep, NULslots, NULsymbols, Nru NULsc, NRUsc, Ns, Nsf, Nf, ru, Qm, RNTI, NNCELLid`
Output	`signal_tx`

Table 6.16 shows the input and output parameters of the function. `RNTI`, `NNCELLid`, `Nf` and `Ns` are used on the scrambling initialization formula. `Qm` shows the modulation scheme to use. `Nsc`, `Nru`, `NRUsc`, `NULsc`, `NULslots` and `NULsymbols` are utilized on the resource mapping of the data and the pilot symbols (`ru`). `NANrep` represents the number of times the only resource unit available should be repeated. In the end, the codeword, `codeword_tx`, is transformed in the signal to be sent, `signal_tx`.

6.2.3.2 Channel Implementation

The **Channel** is represented in Figure 6.1a between the transmitter and receiver parts, where the possible channel models used for testing are equal for NPUSCH formats 1 and 2. Therefore, subsection 6.2.2.2 should be consulted for more information.

6.2.3.3 Receiver Implementation

The **Receiver** is represented in Figure 6.1a by the green dotted lines. The functions inside the gray dotted squares are bypassed, since they are only necessary for the USRP co-simulation testing - section 6.3.

The first steps on the receiver implementation are the general parameters and the DMRS (pilot symbols) generation. These functions are equal on both the transmitter and the receiver sides.

`GetParametersF2()` corresponds to the first function (Table 6.13) and the reason why it is called on both sides is so the simulation works when two different computers are used. Obviously, the input values have to be the same on the transmitter and the receiver. This implementation is accurate, since the eNodeB always informs the UE on what values to use (discussed in section 5.5). When calling this function on the receiver side, the output `HARQACK` is empty.

The second function is called `GenerateDMRSF2()` and its inputs and output are represented on Table 6.15. Afterwards, it is possible to estimate the channel based on the received pilot symbols that are compared with the calculated ones. To do this, function `ChannelEstF2()` is called with its inputs and outputs being represented in Table 6.17.

First, a reference grid is generated using the know pilots (`ru`), with the unknown values being zero. To generate the grid, `NULsc`, `Nsc`, `NULslots`, `NULsymbols` and `NARrep` are neces-

Table 6.17: Function `ChannelEstF2()` description.

Function	`ChannelEstF2()`
Input	`signal_rx, ru, NULsc, Nsc, NULslots, NULsymbols, NANrep`
Output	`H_zf`

sary. Then, `signal_rx` is also rearranged in grid format. Finally, by comparing both grids `H_zf` is calculated, according to subsection 4.3.1.

Function `DemodulationProcessF2()` performs SC-FDMA signal recovery (including resource demapping), demodulation and descrambling according to subsection 4.2.1. Furthermore, it separates the signal in its several repetitions. Each one is demodulated into a matrix with several codewords, one for each repetition.

Table 6.18: Function `DemodulationProcessF2()` description.

Function	`DemodulationProcessF2()`
Input	`signal_rx, H_zf, NULslots, NULsymbols, NANrep, Nru` `NULsc, NRUsc, Nsc, H_zf, Qm, RNTI, NNCELLid, Ns, Nsf, Nf`
Output	`codeword_rx, Ns, Nsf, Nf`

Table 6.18 shows the input and output parameters of the function. `RNTI`, `NNCELLid`, `Nf` and `Ns` are used to perform descrambling. `Qm` is necessary to decide which demodulation scheme to use. `NANrep` is used to know the total number of repetitions. `Nsc`, `Nru`, `NULsymbols`, `NULslots`, `NULsc` and `NRUsc` are important for the resource demapping. `H_zf` is utilized in the equalization step. In the end, each recovered codeword is stored in the matrix, `codeword_rx`. `Ns`, `Nsf` and `Nf` values are updated.

Function `DecodingProcessF2()` performs the decoding step described in subsection 4.2.2, with each codeword in matrix `codeword_rx` being decoded into an `HARQACK` indicator. The values are combined so the most probable indicator is selected. If all of them have errors, the decoding was unsuccessful. All inputs and outputs of the function are displayed on Table 6.19.

Table 6.19: Function `DecodingProcessF2()` description.

Function	`DecodingProcessF2()`
Input	`codeword_rx`
Output	`HARQACK`

The recovered codewords, `codeword_rx`, are displayed when calling the function `DataDispF2()`. The number of slots (`Ns`), subframes (`Nsf`) and frames (`Nf`) utilized is also shown. Inputs of the function are shown on Table 6.20.

6.2.4 NPRACH Simulation

The simulation depicted in Figure 6.1b is divided in transmitter, channel and receiver. Each one of them is explained in depth on sub-subsections 6.2.4.1, 6.2.4.2 and 6.2.4.3.

Table 6.20: Function `DataDispF2()` description.

Function	`DataDispF2()`
Input	`codeword_rx, ns, nsf, nf`
Output	-

6.2.4.1 Transmitter Implementation

The **transmitter** is represented in Figure 6.1b by the blue dotted lines. SIB2 necessary parameters are generated as user input. If the user leaves them blank or chooses an impossible combination of values, they are automatically assigned, when needed.

Table 6.21: Function `NprachGen()` description.

Function	`NprachGen()`
Input	`NNPRACHsc, NNPRACHscoffset, NNPRACHrep, NPRACHFormat, NNCELLid`
Output	`signal_tx`

`NNPRACHsc` represents the total number of subcarriers allocated to NPRACH, `NNPRACHscoffset` corresponds to the frequency location of the first allocated subcarrier and `NNPRACHrep` is the number of times the generated preamble is repeated. `NPRACHFormat` is an user input that defines the CP size - section 3.4. `NNCELLid` has a range of values between 0 and 503. Since its calculation is a downlink procedure and for simplification, this value is always 1 (can easily be changed by the user). The preamble is generated according to section 3.4.

6.2.4.2 Channel Implementation

The **Channel** is represented in Figure 6.1b between the transmitter and receiver implementations, where the possible channel models to be tested are equal to the ones used on NPUSCH formats 1 and 2. Therefore, subsection 6.2.2.2 should be consulted for more information.

6.2.4.3 Receiver Implementation

The **Receiver** is represented in Figure 6.1b by the green dotted lines.

Table 6.22: Function `NprachDetect()` description.

Function	`NprachDetect()`
Input	`signal_rx, NNPRACHsc, NNPRACHscoffset, NNPRACHrep, NPRACHFormat, NNCELLid,`
Output	-

In the receiver side a preamble like the one expected (sent in the transmitter) is generated. Afterwards, a cross-correlation between the received and expected signals is done. If it exceeds a determined threshold, obtained by trial and error, the sequence is detected. It is displayed in the MATLAB editor if it was, in fact, a PRACH sequence or not.

6.3 USRP Co-simulation

This section describes the implementation of a RF front-end (USRP) that transmits the baseband signal, previously generated on MATLAB. Furthermore, another USRP captures the transmitted signal to be decoded on MATLAB. This co-simulation environment allows the generation/capture of signals in real-time with actual hardware, by reusing the MATLAB behavioral model. This has two advantages. First, a closer to reality implementation allows to further test the protocol and to correct possible bugs, which helps when an implementation of the protocol is made in actual hardware. Secondly, it gives a degree of comparison between the simulation-only environment, where no time constrains need to be met.

The chosen platform was an USRP B200 (only one input/output) for the transmitter and a B210 (two inputs/two outputs) for the receiver. A USRP is a Software Defined Radio (SDR) system, which consists on a radio transceiver, where components usually implemented in hardware are instead implemented by means of software. This reduces development costs drastically. This is the first reason why the USRP was chosen. Two more reasons were considered. First, the USRP package available on MATLAB greatly simplifies the interfacing between the laptop and the RF front-end. Secondly, contrary to some SDR devices, the connection to the laptop is made using Universal Serial Bus (USB) 3.0, which increases the reliability and the speed of the connection. The choice between B200 and B210 is due to the number of inputs/outputs, since NB-IoT is SISO or SIMO on uplink.

Observing Figure 6.1a, it's possible to conclude two new blocks need to be added on the receiver side of the simulation. Their main goal is to synchronize the received and transmitted signals. This synchronization consists of two major parts: Carrier Frequency Offset (CFO) and Symbol Time Offset (STO). This synchronization at the receiver must be performed.

More details about the USRP implementation can be found on subsection 6.3.1. More information about CFO and STO synchronization functions is described in subsections 6.3.2 and 6.3.3, respectively.

6.3.1 USRP Implementation

The USRP transmission/reception requires several steps, depicted in Figure 6.2.

The blue and greed dotted lines represent the transmitter and receiver simulated/implemented in MATLAB, shown in Figure 6.1a. The output of the MATLAB transmitter (`signal_tx`) is in baseband format. The USRP receives this waveform from the laptop, via a USB 3.0 connection, with 16 bits of resolution.

On the actual USRP, there is a Field-Programmable Gate Array (FPGA) that performs the Digital Signal Processing (DSP) operations. That part is represented in Figure 6.2 in orange. The other operations are done in the USRP RF daughter-board, the gray part of Figure 6.2.

Both I/Q components of the complex baseband signal are up-converted on the USRP's FPGA. Then, they are converted to analog format, go through a filter and are amplified. This is done in the USRP RF daughter-board. Afterwards, the signal is transmitted using a Delock's omni-directional SubMiniature version A (SMA) antenna (-3.5dBi at 900MHz).

On the receiver side, the chain is inverted. The signal goes from the RF format into baseband. First, it is amplified and divided in I/Q components. Those components are filtered and go through an Analog-to-Digital Converter (ADC). At this point, the signal is in digital format and ready to enter the USRP's FPGA to be down-converted. The I/Q

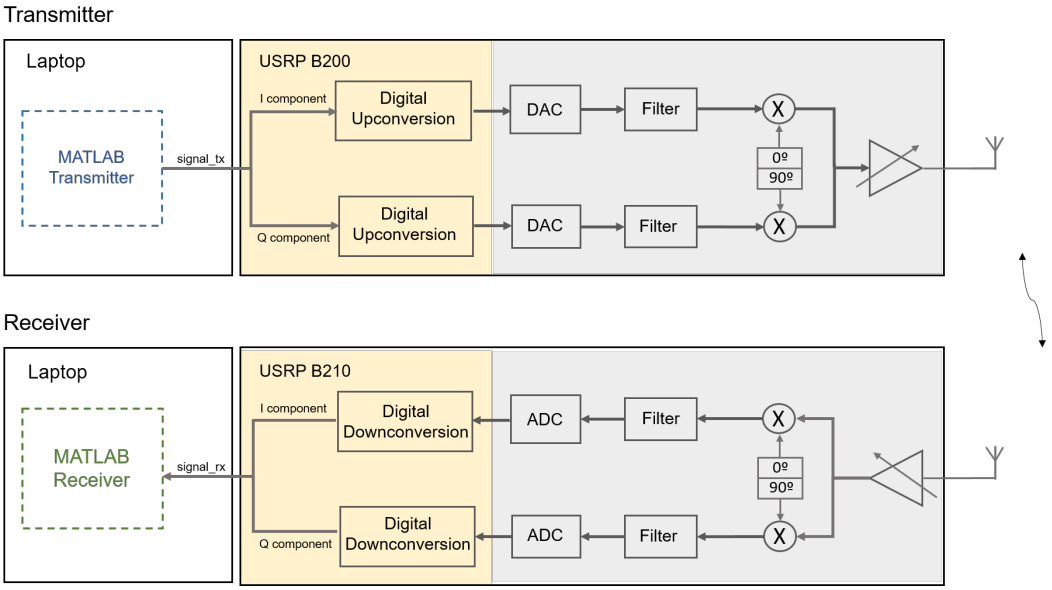


Figure 6.2: Simplified model of the USRP modulation/demodulation process [oE17].

components are sent to the laptop using 16 bit resolution. The baseband signal (`signal_rx`) enters the MATLAB receiver and goes through the simulation chain, represented in Figure 6.1a. In this simulation, the functions inside the gray dotted squares have to be included, since they are necessary to perform CFO and STO compensations.

To configure the USRP components it was used the MATLAB USRP Support Package. It includes the Universal Hardware Driver (UHD) driver and provides a design and modeling environment. The first step is to define properties of a transmitter and receiver object.

When calling the `'transmitter_object = comm.SDRuTransmitter'` MATLAB function, available on the USRP package, a transmitter object is created. The following `transmitter_object` properties should be defined:

1. "CenterFrequency" - the center frequency used is 900MHz (for stand-alone mode only).
2. "Gain" - RF front-end gain used is 25dB.
3. "MasterClockRate" - The D/A clock rate is (1.92×4MHz). Valid ranges are between 5^6 and 56^6 for B200 and B210 USRPs.
4. "InterpolationFactor" - Interpolation value is 4, which corresponds to a sampling rate of MasterClockRate/4, or approximately, 1.92MHz (value stated in the protocol specs).
5. "TransportDataType" - Transport data type used is 'int16'.

Afterwards, the object can accept an `input signal` from MATLAB, transmitting it to the board using the UHD - function `'step(transmitter_object, input_signal)'` is used. In this case, `input_signal` is a vector of double precision [Mat17b].

When calling the `'receiver_object = comm.SDRuReceiver'` MATLAB built-in function, a receiver object is created. The following `receiver_object` properties should be defined:

1. “CenterFrequency” - the center frequency used is 900MHz (for stand-alone mode only).
2. “Gain” - RF front-end gain used is 30dB.
3. “MasterClockRate” - The A/D clock rate is (1.92×4MHz). Valid ranges are between 5^6 and 56^6 for B200 and B210 USRPs.
4. “DecimationFactor” - Decimation value is 4, which corresponds to a sample rate of MasterClockRate/4, or approximately, 1.92MHz (value stated in the protocol specs).
5. “TransportDataType” - Transport data type used is ‘int16’.
6. “FrameLength” - Length of each frame to be received.
7. “NumFramesInBurst” - Number of frames received in a row, or in other words, the expected signal length.

Function ‘`[output_signal, LEN] = step(receiver_object)`’ receives signals from the USRP board using UHD packets. `output_signal` is a vector of complex double precision. `LEN` is the actual data length [Mat17a].

6.3.2 Carrier Frequency Offset

Function `frequencyoffset()` addresses the problem of the frequency mismatch between the transmitter and the receiver. The algorithm is based on the added CP that precedes each SC-FDMA symbol.

According to [JKD16], the estimated CFO can be calculated from the ‘argument’ of the multiplication of the SC-FDMA symbol conjugate by its CP according to equation 6.1:

$$\xi = \frac{1}{2pi} \arg\left(\sum_{n=1}^{N_{CP}} r^*(n - N_{CP} + N_{IFFT})C(n)\right) \tag{6.1}$$

where *r(n)* is the received symbol without CP, with N_{IFFT} being its length. N_{CP} is the CP length with its actual values being *C(n)*.

This is done for each received symbol. Then, the several estimated values are averaged into a final CFO value.

Table 6.23 shows the inputs and outputs of the previously explained function. `NULsc` is necessary because symbol length (N_{IFFT}) varies with it. `data_rxt` is the CFO compensated version of `data_rxft`.

Table 6.23: Function `FrequencyOffset()` description.

Function	`FrequencyOffset()`
Input	`data_rxft, NULsc`
Output	`data_rxt`

Since the allowed range of values for the function ‘argument’ is $[-\pi, \pi]$, the estimated normalized CFO range is $[-0.5, 0.5] \Rightarrow [-\frac{\Delta f}{2}, \frac{\Delta f}{2}]$. Considering CFO can be larger, this technique can lead to errors in the final estimated value.

6.3.3 Symbol Time Offset

Function `TimeOffset()` addresses the problem of the unknown SC-FDMA symbol arrival time. As previously stated (in section3.3), the pilot symbols (DMRS) are known at the receiver and so is their position on the resource grid. Depending on the `Format`, `ru` is calculated with the help of `NNCELLid`, `Nru`, `NULslots` and `NRUsc`. Then, these symbols are put in a grid format (the unknown values are zero) and go through the SC-FDMA modulation chain, including repetitions. All the remaining parameters are necessary to perform this step. Depending on `Format` value, the set of variables required to use vary. The ones not used are empty matrices. This way, it is obtained a signal that can be compared with the received one. By using the built-in MATLAB function `'xcorr'`, it's obtained the cross-correlation sequence between the known values signal with the received one. The maximum correlation index value corresponds to the symbol time offset.

Table 6.24: Function `TimeOffset()` description.

Function	`TimeOffset()`
Input	`data_rxt, Format, NRUsc NNCELLid, Nru, NULslots, NULsc, deltaf, Nsc, NULsymbols, Qm, MNPUSCHrep, MNPUSCHidentical, NANRep, Nslots`
Output	`data_rx`

The number of available DMRS may not be enough to give an accurate time offset. Therefore, fine synchronization might be necessary in order to obtain an exact STO value.

This chapter explained the implemented practical work. It is divided in two main parts. The first one is the MATLAB simulation, where transmitter/receiver chains were coded. After, a co-simulation using two USRPs was described. On the next chapter, both implementations are evaluated. Constellations, eye diagrams, Bit Error Ratio (BER) performances, PAPR analysis and magnitude spectrums are obtained and analyzed. A comparison between both parts of the practical work is provided.

Chapter 7

Results

This chapter is focused on the overall performance results. Both MATLAB simulation environment and USRP co-simulation are subjected to testing and analysis. Section 7.1 is focused on the MATLAB simulation and section 7.2 on the USRP co-simulation results.

7.1 Simulation Results

Simulation results for NPUSCH format 1, NPUSCH format 2 and NPRACH are displayed in this section. Magnitude spectrums, constellations, eye diagrams and BER performances are measured using different channel models (AWGN and fading channel). Performance with and without ZF equalizer/turbo coding is compared. Furthermore, different modulation schemes' (BPSK and QPSK) performance is also analyzed. Finally, NPRACH preamble is studied, since by being a ZC sequence its CAZAC properties should be verified.

Appendix A is a small user guide, that exemplifies how to run all the possible simulations. Section A.1 explains how the run NPUSCH format 1 and 2 simulations, section A.2 shows how to run the BER measure simulation and section A.3 describes how to run the NPRACH simulation.

7.1.1 NPUSCH Format 1 Simulation Results

This subsection displays the results regarding NPUSCH format 1. Figure 7.1 presents a schematic of the implemented chain. It is divided in coding/decoding procedures, which is in blue color, and modulation/demodulation procedures, in green color. Points where constellations and eye diagrams were obtained are represented by the letter a) and b). The first corresponds to the results on the transmitter side and the second on the receiver side. Several switches are represented throughout the figure, using points 1), 2) and 3). 1) and 2) are used to measure BER results. Essentially, those parts were bypassed on the code, so the BER gain could be measured with and without that specific step. This way, the gain difference could be obtained and, as a consequence, their importance in the general chain could be assessed. Point 1) corresponds to the turbo coder/decoder bypass and 2) means no equalizer is used. Point 3) is in Figure 7.1 to represent the several channels that can be tested. Each `mode` corresponds to a different channel model. More information about each mode can be found on sub-subsection 6.2.2.2. Finally, in the blue circles are the transmitted and received transport blocks. When compared, the BER can be calculated. In appendix

B.1, a known sequence was imposed on the beginning of the chain so a reference sequence test is available in each main point of the implemented simulation.

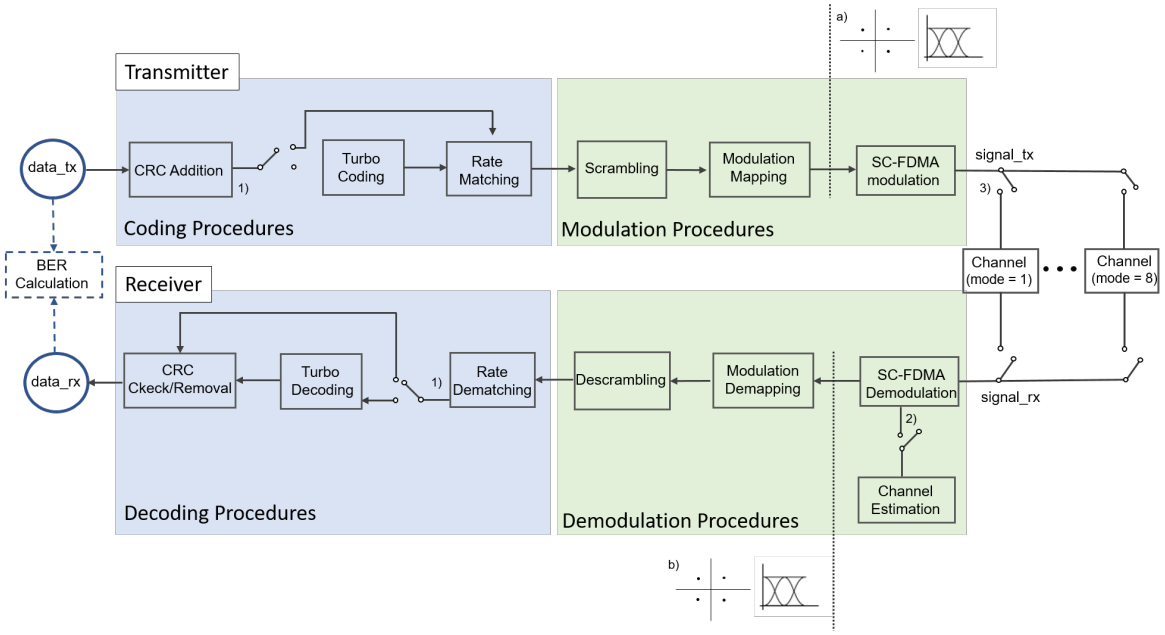


Figure 7.1: MATLAB implemented chain schematic for NPUSCH format 1. Measures of constellations and eye diagrams happen on points a) and b).

Several measures are taken and analyzed throughout this subsection. A small overview of them is made below:

1. On sub-section 7.1.1.1, it is displayed the transmitted constellation and eye diagram, for both BPSK and QPSK. These results were obtained on point a) of Figure 7.1.

2. On sub-section 7.1.1.2, it is shown the received constellations and eye diagrams, when using an AWGN channel - on point 3) of Figure 7.1, `mode` = 1. Two different E_b/N_0 values were tested - 0dB and 8dB. Only QPSK modulation scheme was analyzed. These results were obtained on point b) of Figure 7.1.

3. On sub-section 7.1.1.3, it is depicted the received constellations and eye diagram, when using 3GPP specified fading channel models. Three fading models were tested (EPA, EVA, ETU) - on point 3) of Figure 7.1, `mode` = 2,3,4. Only the QPSK modulation scheme was analyzed. These results were obtained on point b) of Figure 7.1.

4. On sub-section 7.1.1.4, it is presented the received constellations and eye diagrams, when using 3GPP specified fading channel models with added AWGN. Only the EVA model was tested - on point 3) of Figure 7.1, `mode` = 6. Both QPSK and BPSK modulation schemes were evaluated. Two different E_b/N_0 values were analyzed - 0dB and 8dB. These results were obtained on point b) of Figure 7.1.

5. On sub-section 7.1.1.5, it is analyzed the BER performance, when using the AWGN channel mode - on point 3) of Figure 7.1, `mode` = 1. Tests with and without turbo coding/equalizer for both QPSK and BPSK were analyzed - switches 1) and 2) of

Figure 7.1. Afterwards, BER performances for the three channel fading models with AWGN were tested (on point 3) of Figure 7.1, `mode` = 5,6,7), with both switches closed. Then, an analysis when using the EPA channel model with AWGN (on point 3 of Figure 7.1, `mode` = 6) with and without turbo coding/equalizer for both QPSK and BPSK was made - switches 1) and 2) of Figure 7.1. Finally, the influence of N_{Rep} on the BER value was tested.

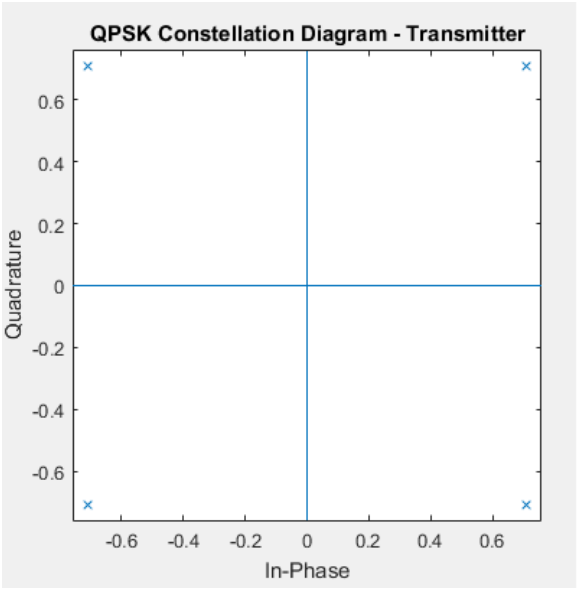
6. On sub-section 7.1.1.6, it is displayed the magnitude spectrum of the transmitted and received signals .

7. On sub-section 7.1.1.7, it is calculated the PAPR value of the SC-FDMA transmission.

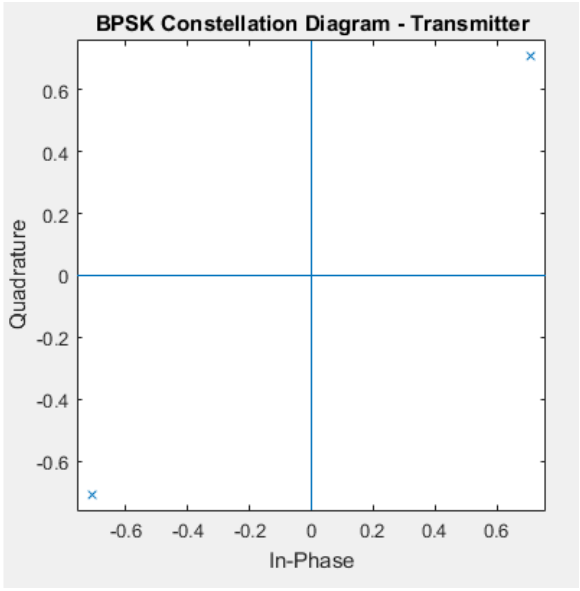
Values used when coding and modulating the transport block are: Δf = 15kHz, RNTI = 1, I_{sc} = 5, I_{MCS} = 2, I_{RU} = 2, I_{Rep} = 3, rv_{dci} = 0 for QPSK and Δf = 15kHz, RNTI = 1, I_{sc} = 5, I_{MCS} = 0, I_{RU} = 2, I_{Rep} = 3 and rv_{dci} = 0 for BPSK. The only value that changes is the one that selects the modulation to be used, so an accurate comparison between BPSK and QPSK can be made. These values are constant throughout this subsection, unless otherwise mentioned.

7.1.1.1 Transmitter - Constellations and Eye Diagram

Figure 7.2 shows the transmitted constellation. It is obtained using the available MATLAB function `'scatterplot'`, after the modulation mapping step is performed - point a) of Figure 7.24. Results are displayed both for QPSK (Figure 7.2a) and BPSK (Figure 7.2b) modulation schemes.



(a) QPSK transmitter constellation.



(b) BPSK transmitter constellation.

Figure 7.2: Transmitter constellations.

Figure 7.3 shows the transmitted eye diagram. It is obtained using the built-in MATLAB function `'eyediagram'` on point a) of Figure 7.24.

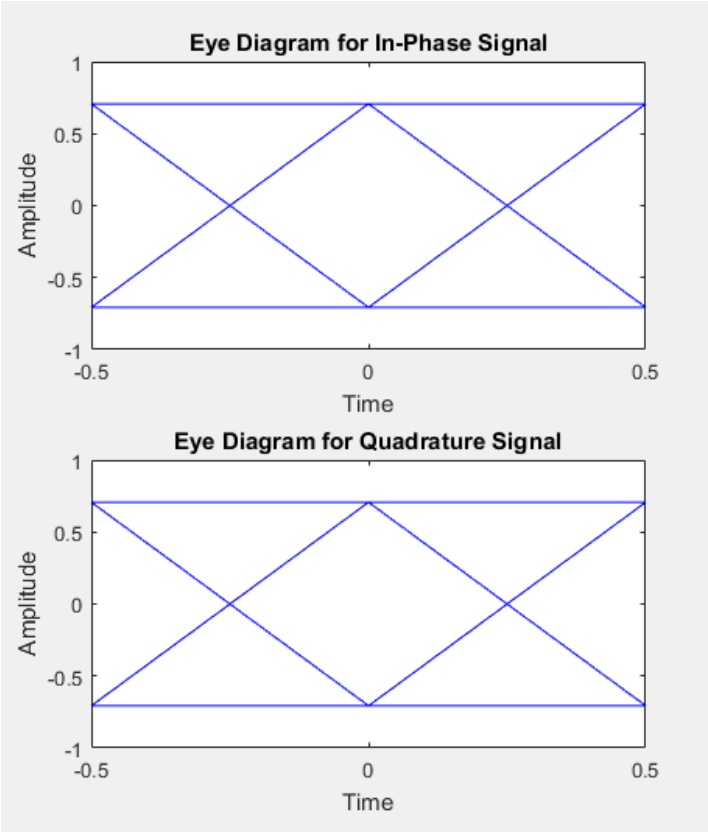


Figure 7.3: Transmitter eye diagram - BPSK/QPSK have equal eye diagrams.

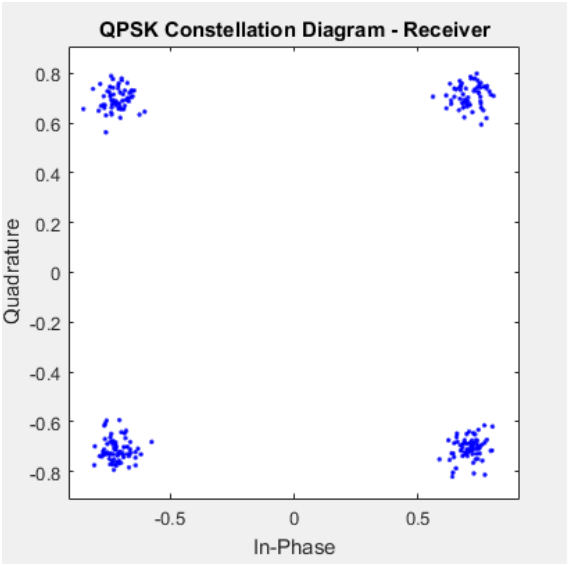
When comparing the MATLAB constellations with the expected ones, depicted on Figures 3.8 and 3.9, it is easy to conclude results are according to expected.

7.1.1.2 AWGN Channel Model - Constellations and Eye Diagrams

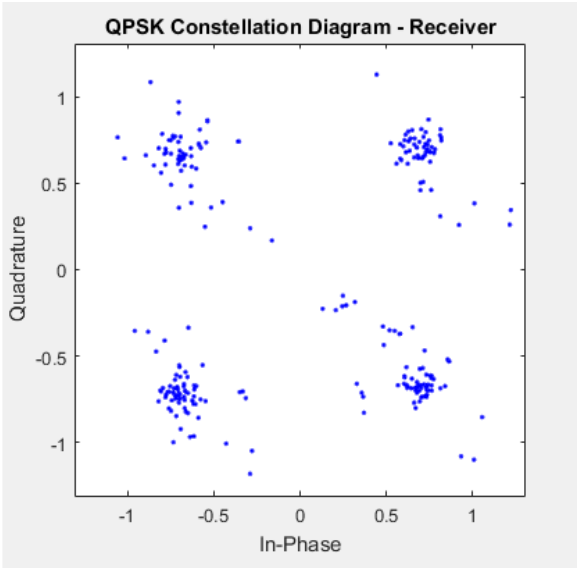
This sub-subsection results were obtained considering the channel is only composed of AWGN - according to sub-subsection 6.2.2.2, this means that on point 3) of Figure 7.24, `mode` = 1. Two values of E_b/N_0 were tested. Only QPSK results are considered at this stage to avoid repetition.

First, results for E_b/N_0 equal to 8dB are depicted. Figure 7.4 shows the constellation with and without equalizer and Figure 7.5 shows the eye diagram with and without equalizer. Secondly, results for E_b/N_0 equal to 0dB are depicted. Figure 7.6 shows the constellation with and without equalizer and Figure 7.7 shows the eye diagram with and without equalizer.

It is possible to conclude, the results with equalizer are considerably worst. This is due to the fact that ZF equalizers do not compensate additive noise. As a consequence, any added noise gets boosted by a large factor which destroys the actual signal. Furthermore, when $E_b/N_0 = 0$dB, the signal can only be recuperated if no equalizer is utilized.

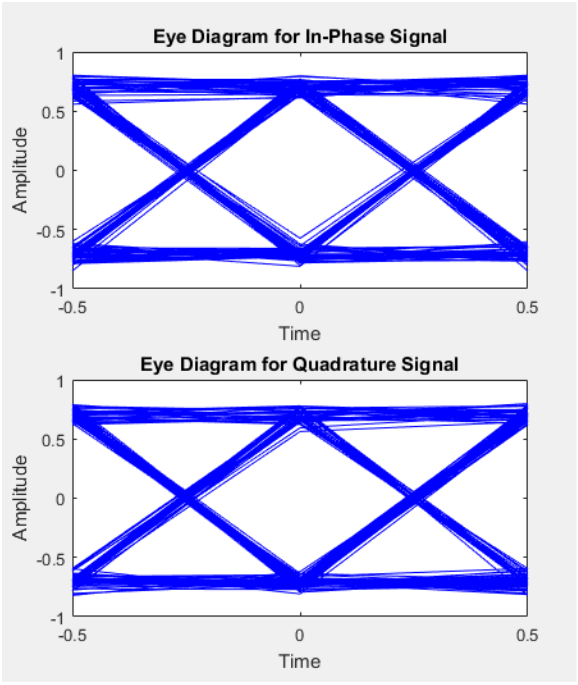


(a) QPSK constellation for AWGN channel with no equalizer present.

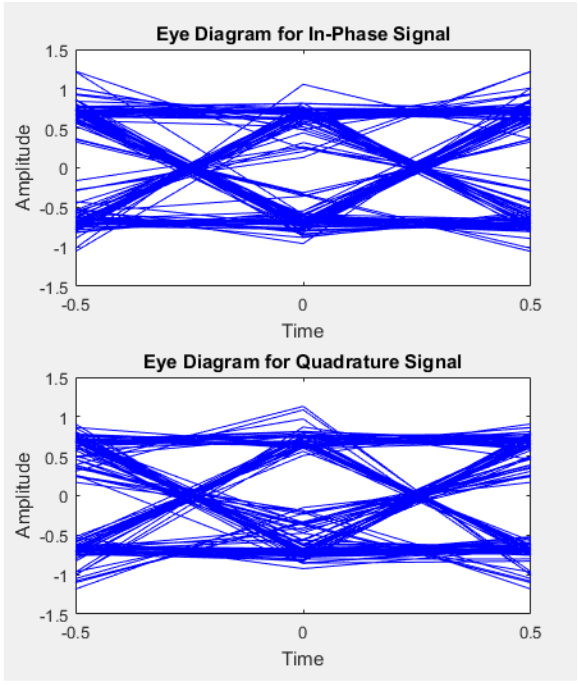


(b) QPSK constellation for AWGN channel when using ZF equalizer.

Figure 7.4: QPSK constellation for AWGN channel when E_b/N_0 is 8dB.

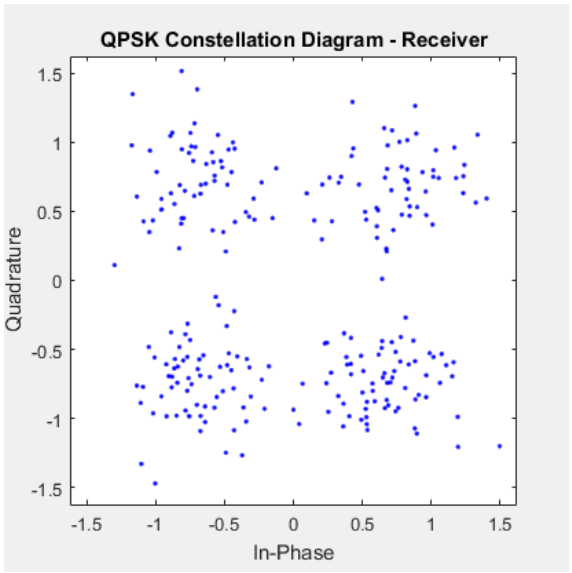


(a) QPSK eye diagram for AWGN channel with no equalizer present.

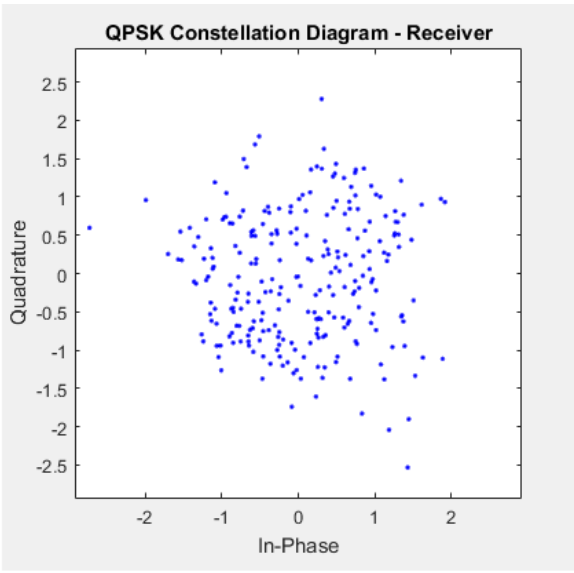


(b) QPSK eye diagram for AWGN channel when using ZF equalizer.

Figure 7.5: QPSK eye diagram for AWGN channel when E_b/N_0 is 8dB.

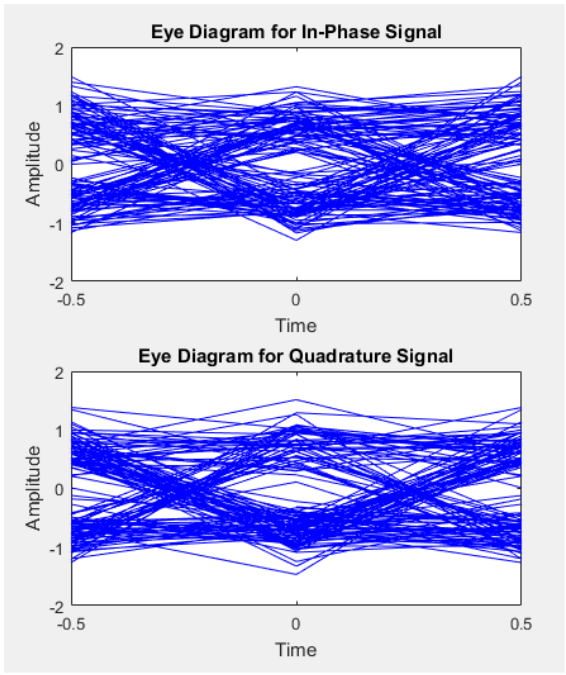


(a) QPSK constellation for AWGN channel with no equalizer present.

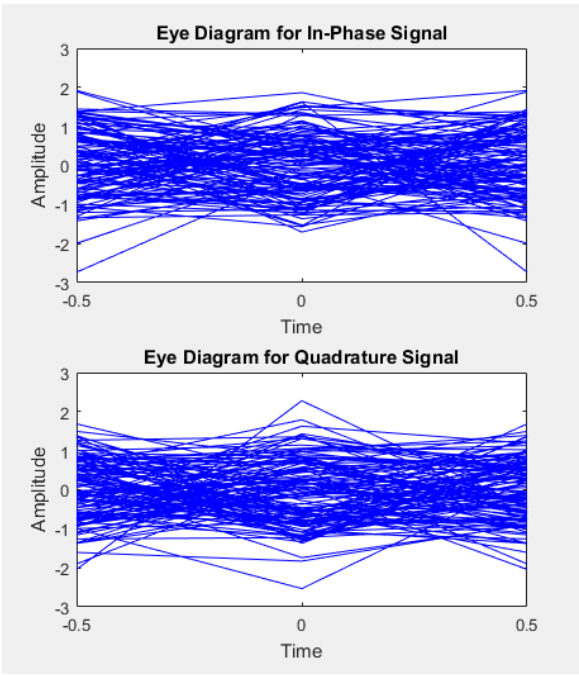


(b) QPSK constellation for AWGN channel when using ZF equalizer.

Figure 7.6: QPSK constellation for AWGN channel when E_b/N_0 is 0dB.



(a) QPSK eye diagram for AWGN channel with no equalizer present.



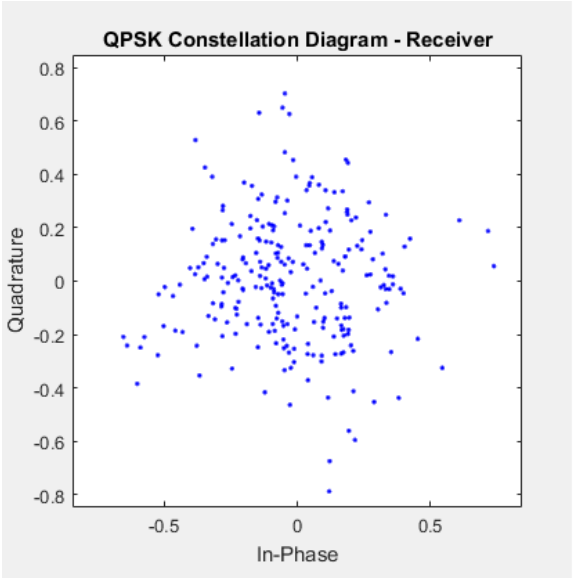
(b) QPSK eye diagram for AWGN channel when using ZF equalizer.

Figure 7.7: QPSK eye diagram for AWGN channel when E_b/N_0 is 0dB.

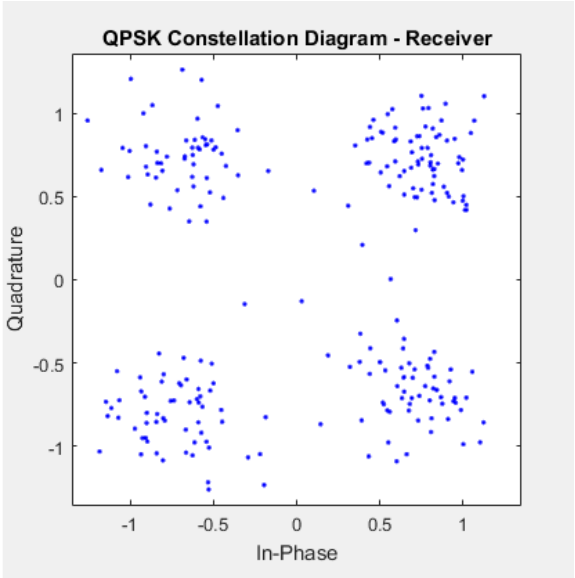
7.1.1.3 Fading Channel Models - Constellations and Eye Diagrams

This subsection considers only the fading channel models that do not include AWGN - on point 3) of Figure 7.24, `mode` = 2/3/4. Only QPSK results are considered at this stage to avoid repetition.

First, results for EPA channel model (Table 6.5) are depicted. Figure 7.8 shows the constellation with and without equalizer and Figure 7.9 represents the eye diagram with and without equalizer. After, results for EVA channel model (Table 6.6) are depicted. Figure 7.10 shows the constellation with and without equalizer, and Figure 7.11 represents the eye diagram with and without equalizer. Finally, results for ETU channel model (Table 6.7) are depicted. Figure 7.12 shows the constellation with and without equalizer, and Figure 7.13 represents the eye diagram with and without equalizer.

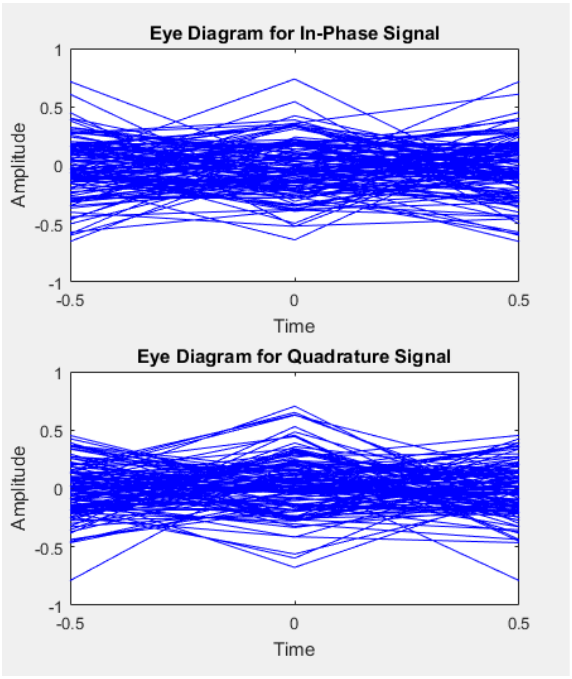


(a) QPSK constellation for EPA channel model with no equalizer present.

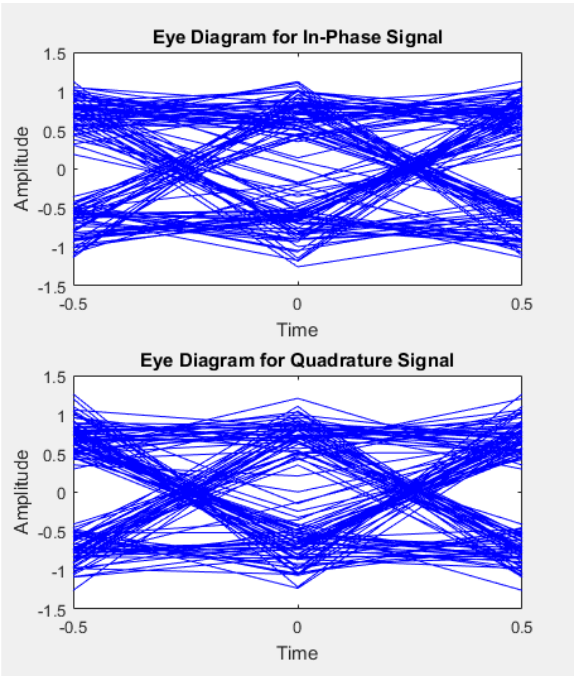


(b) QPSK constellation for EPA channel model when using ZF equalizer.

Figure 7.8: QPSK constellation for EPA channel model.

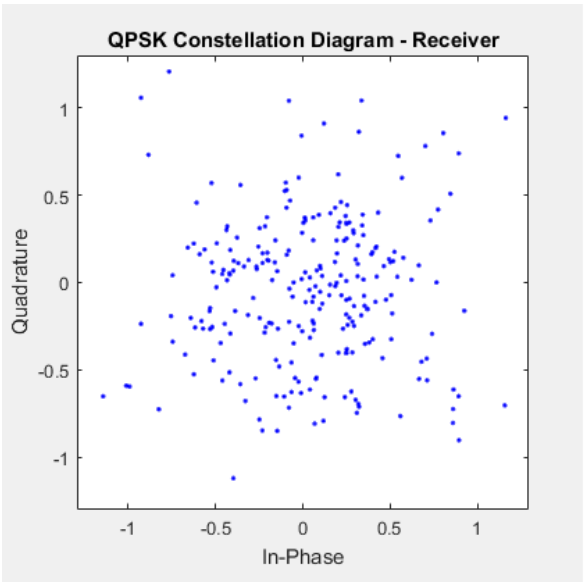


(a) QPSK eye diagram for EPA channel model with no equalizer present.

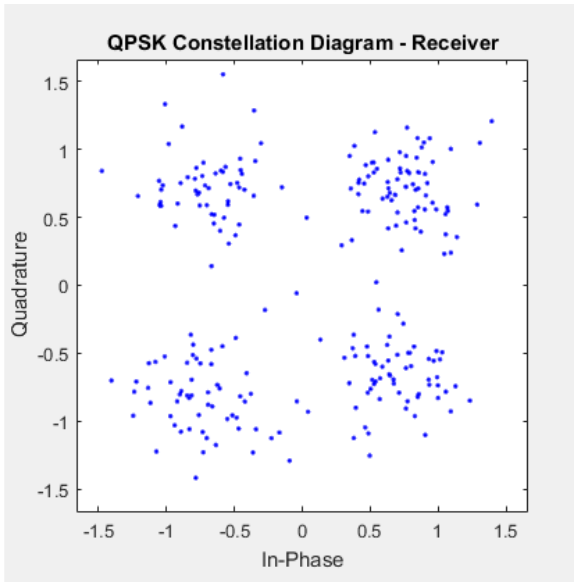


(b) QPSK eye diagram for EPA channel model when using ZF equalizer.

Figure 7.9: QPSK eye diagram for EPA channel model.

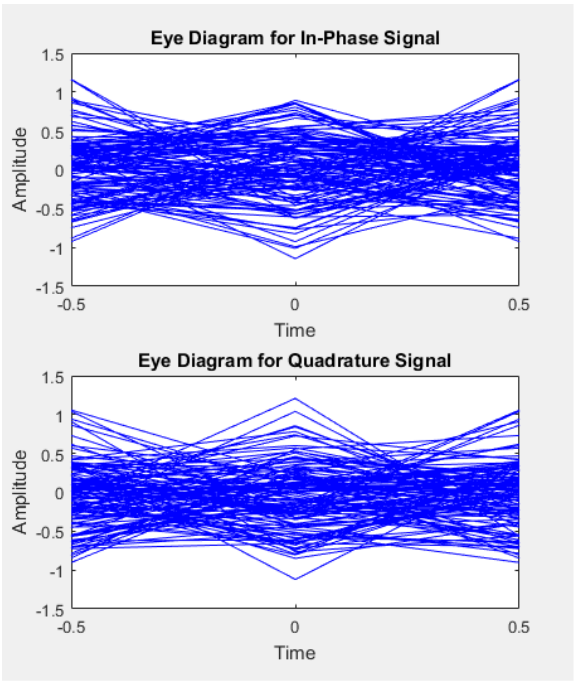


(a) QPSK constellation for EVA channel model with no equalizer present.

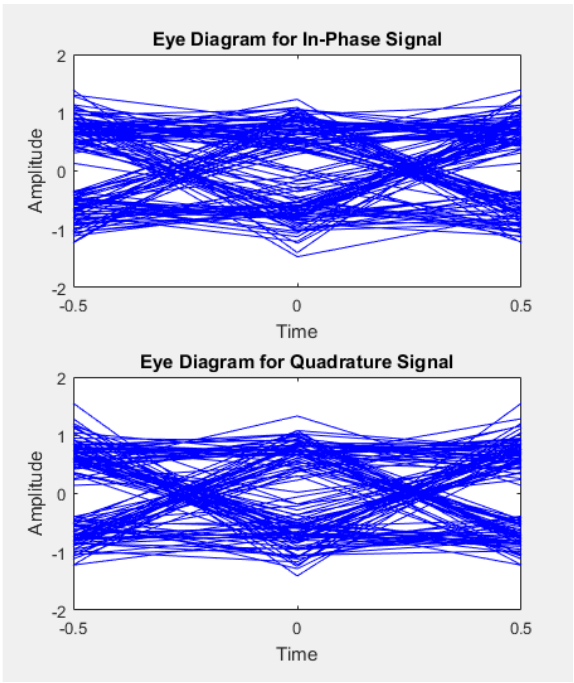


(b) QPSK constellation for EVA channel model when using ZF equalizer.

Figure 7.10: QPSK constellation for EVA channel model.

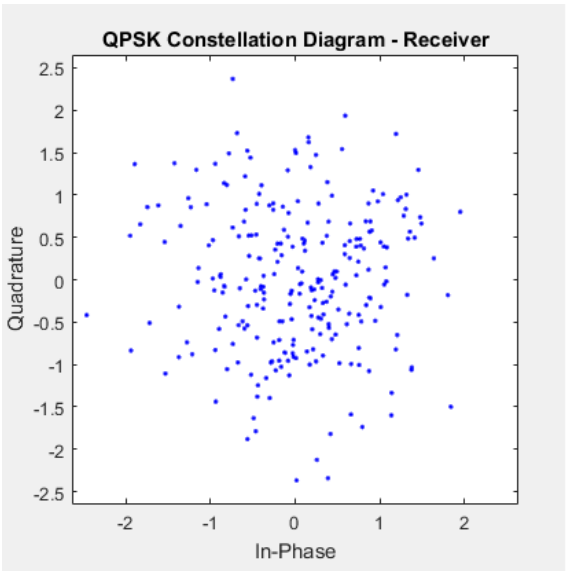


(a) QPSK eye diagram for EVA channel model with no equalizer present.

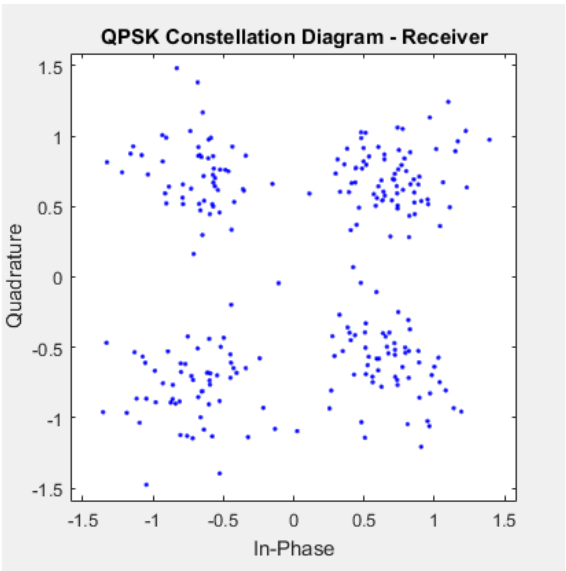


(b) QPSK eye diagram for EVA channel model when using ZF equalizer.

Figure 7.11: QPSK eye diagram for EVA channel model.

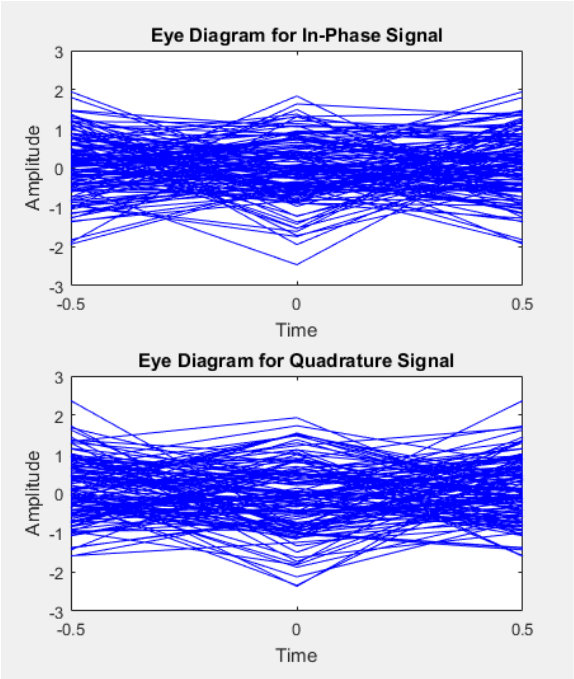


(a) QPSK constellation for ETU channel model with no equalizer present.

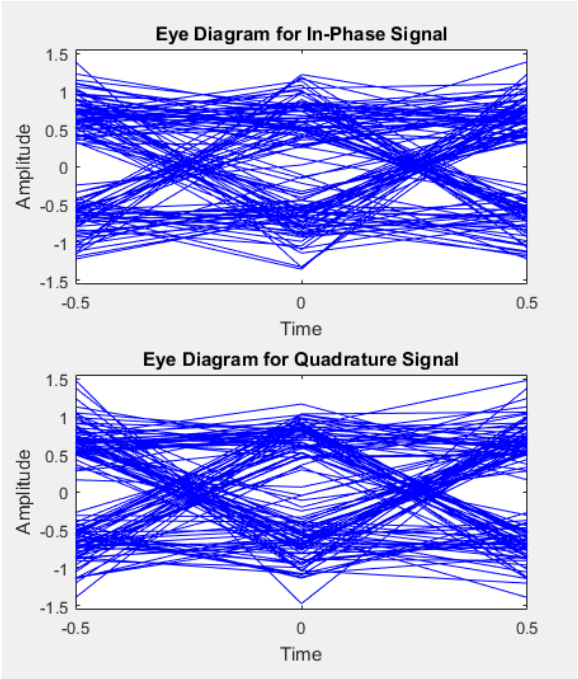


(b) QPSK constellation for ETU channel model when using ZF equalizer.

Figure 7.12: QPSK constellation for ETU channel model.



(a) QPSK eye diagram for ETU channel model with no equalizer present.



(b) QPSK eye diagram for ETU channel model when using ZF equalizer.

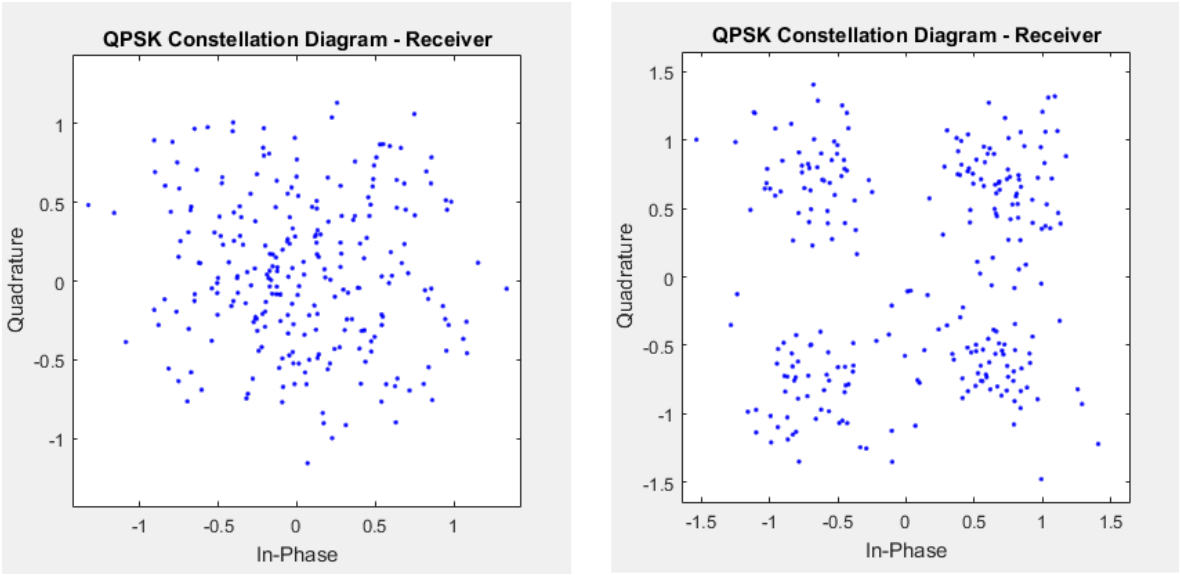
Figure 7.13: QPSK eye diagram for ETU channel model.

It is possible to conclude, the results with equalizer are considerably better, which is expected. Therefore, in the three channel models the signal is always recovered, as long as ZF equalizer is used.

7.1.1.4 Fading Channel with AWGN - Constellations and Eye Diagrams

This sub-subsection considers the channel is composed by a fading channel model (EVA) with AWGN - in point 3) of Figure 7.24, this means `mode` = 6. Both QPSK and BPSK results are considered at this stage to provide some degree of comparison. Only EPA channel model is considered to avoid repetition. E_b/N_0 is equal to 8dB.

QPSK: First, QPSK is considered. Results are depicted in Figure 7.14, which shows the constellation with and without equalizer, and Figure 7.15, which shows the eye diagram with and without equalizer.

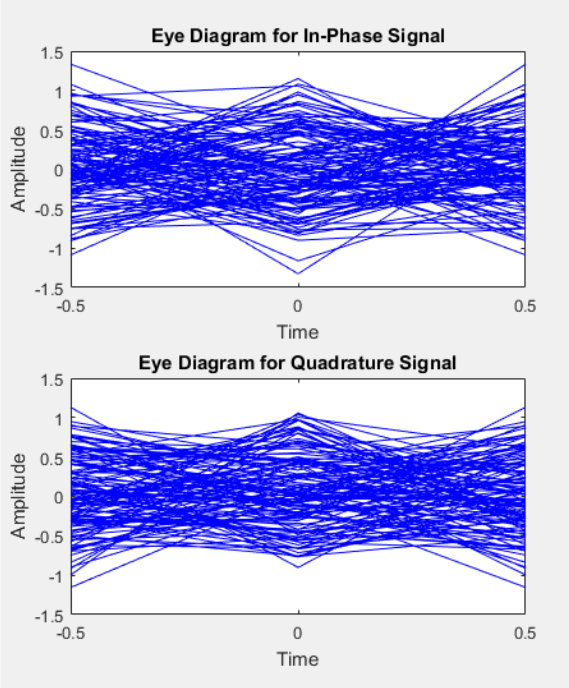


(a) QPSK constellation for ETU channel model with no equalizer present.

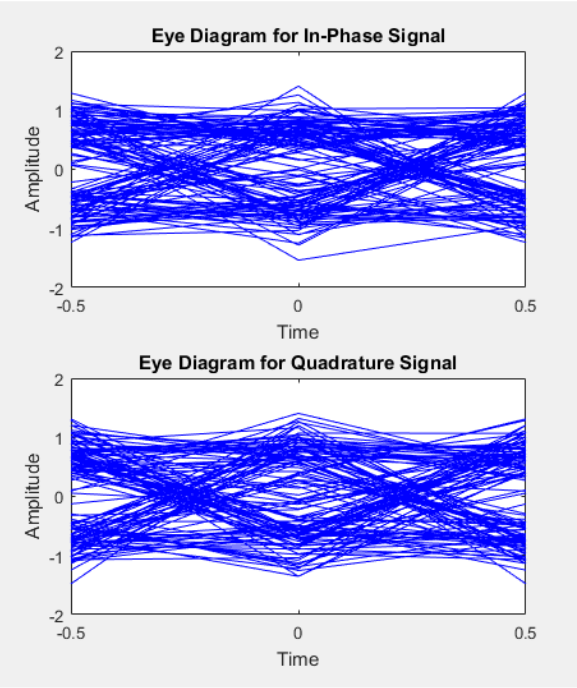
(b) QPSK constellation for ETU channel model when using ZF equalizer.

Figure 7.14: QPSK constellation for EVA channel model with AWGN (E_b/N_0 is 8dB).

BPSK: After, BPSK is considered. Figure 7.16 shows the constellation with and without equalizer, and Figure 7.17 shows the eye diagram with and without equalizer.

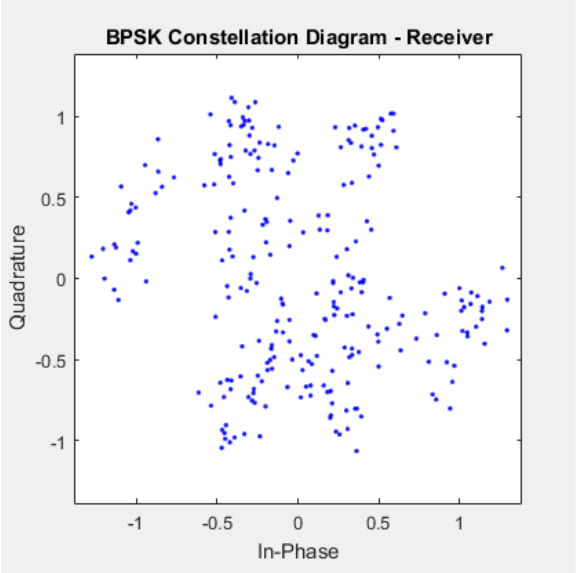


(a) QPSK eye diagram for ETU channel model with no equalizer present.

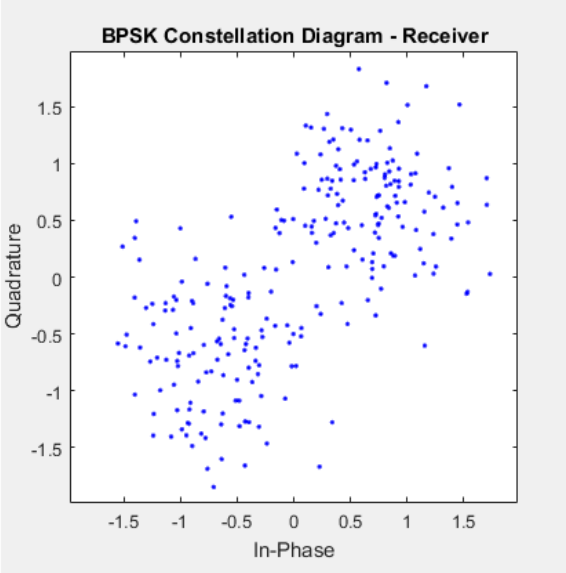


(b) QPSK eye diagram for ETU channel model when using ZF equalizer.

Figure 7.15: QPSK eyediagram for EVA channel model with AWGN (E_b/N_0 is 8dB).

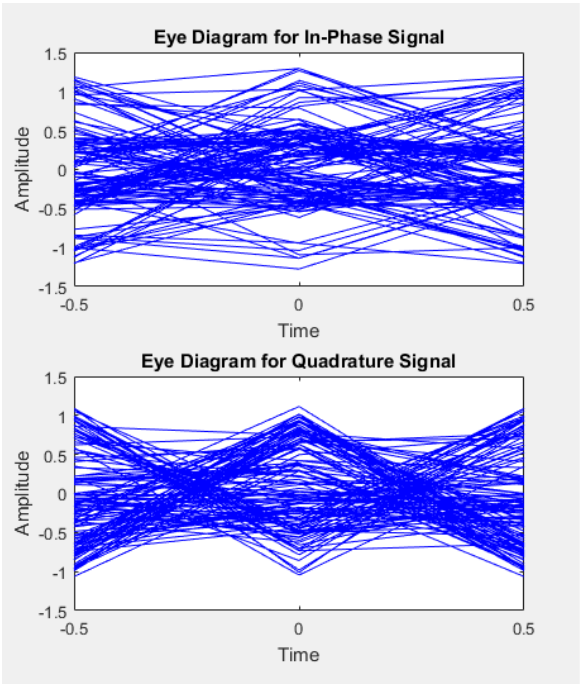


(a) BPSK constellation for ETU channel model with no equalizer present.

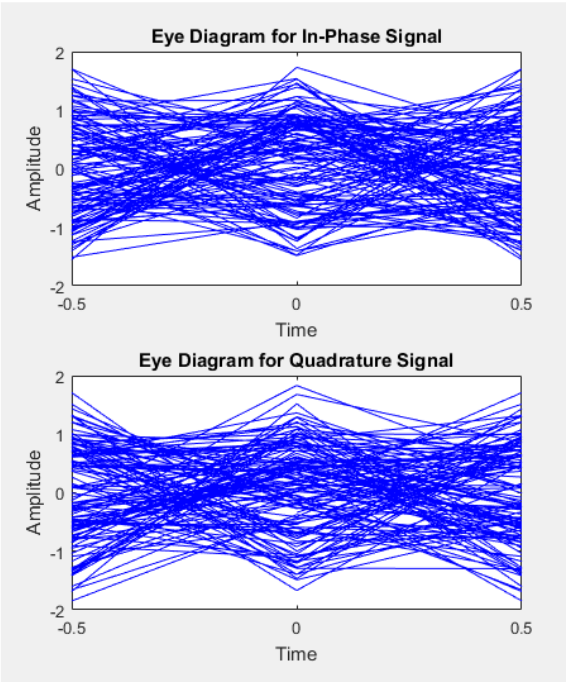


(b) BPSK constellation for ETU channel model when using ZF equalizer.

Figure 7.16: BPSK constellation for EVA channel model with AWGN (E_b/N_0 is 8dB) - no equalizer is implemented on the left figure and ZF equalizer is used on the right figure.



(a) BPSK eye diagram for ETU channel model with no equalizer present.



(b) BPSK eye diagram for ETU channel model when using ZF equalizer.

Figure 7.17: BPSK eye diagram for EVA channel model with AWGN (E_b/N_0 is 8dB) - no equalizer is implemented on the left figure and ZF equalizer is used on the right figure.

It is possible to conclude, the results with equalizer are considerably better, which is expected. Both signals with E_b/N_0 equal to 8dB, are possible to recuperate. For lower values of E_b/N_0, where more AWGN is present, the signal won't be possible to decode. This means a equalizer with better performance (which takes AWGN into consideration), should be used if the situation requires it.

To conclude, the signal is always recovered, as long as ZF equalizer is used and E_b/N_0 isn't lower then 8dB. This value is obviously variable, depending on the N_{Rep} value and the number of turbo decoding iterations as explained in subsections 7.1.1.5 and 4.1.2.2, respectively.

When comparing with the results of sub-subsection 7.1.1.2, this are slightly worst for the same E_b/N_0. Therefore, even though the equalizer compensates the fading component, its effects are still noticeable.

When comparing both modulation schemes, even though QPSK has double bit rate for the same bandwidth, both performances are quite similar in the tested conditions.

7.1.1.5 BER

The BER corresponds to the ratio between the number of wrong bits in the received data with the total number of bits.

Performing a BER simulation is a lengthy process. It's necessary to run individual simulations at each E_b/N_0 of interest. The results also have to be statistically significant. For example, a BER of 10^{-5} means one bit in 100 000 bits will have an error. If the measured

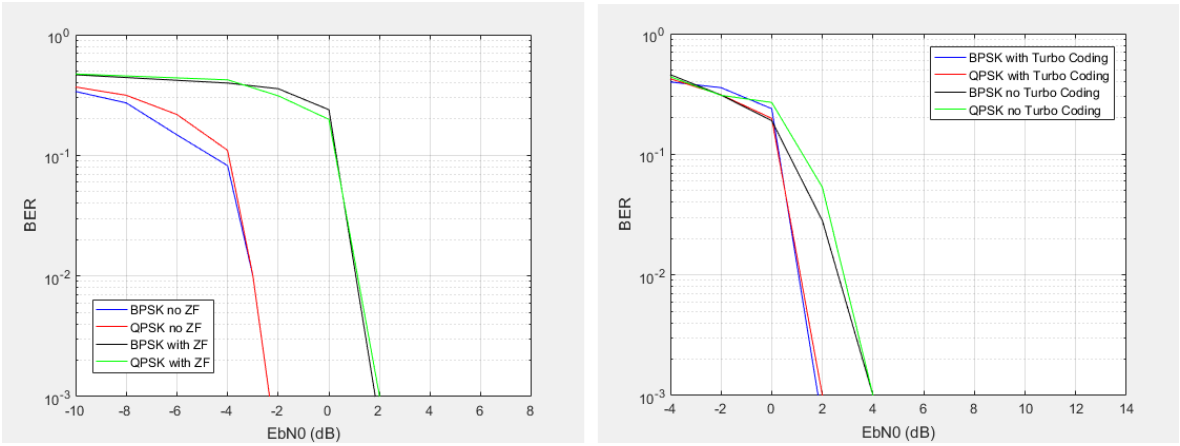
data only contains 100 bits, the error probably won't be seen.

Once enough simulations are performed for each E_b/N_0 value of interest, it's possible to plot the results. The Y-axis should be plotted on a logarithmic scale, whereas the X-axis should be plotted on a linear scale [Gil03]. This requires the MATLAB function `'semilogy'` instead of the more common `'plot'`.

In all simulations, BER tests were run 1000 times for each E_b/N_0. Performance was evaluated for an AWGN channel, between the three fading channel models under test (EPA, EVA and ETU) and for one of the fading channel models with added AWGN.

BER - AWGN channel model: In this subsection, the BER is calculated considering an AWGN channel. Performances with and without turbo encoder/equalizer are considered - switches 1) and 2) of Figure 7.1. QPSK and BPSK implementations are compared.

Figure 7.18a shows the BER for both QPSK and BPSK implementations with and without equalizer. Figure 7.18b depicts the BER performances for both QPSK and BPSK modulation schemes with and without turbo encoding.



(a) BER performance results for an AWGN channel with and without equalizer for both QPSK and BPSK.

(b) BER performance results for an AWGN channel with and without turbo encoding for both QPSK and BPSK.

Figure 7.18: BER performance results for an AWGN channel.

Observing Figure 7.18, one might conclude that there is no advantage to BPSK, since in both examples it performs identically to QPSK, but has half the bit rate.

Even though BPSK and QPSK have the same BER performance, the Signal-to-Noise Ratio (SNR) will differ by 3dB for the same BER. This is why BPSK is still useful. Equation 7.1 represents the general SNR formula and 7.2 shows the relation between QPSK and BPSK bit rates.

$$SNR = \frac{R_b \times E_b}{B \times N_0} \tag{7.1}$$

$$R_b(QPSK) = 2 \times R_b(BPSK) \tag{7.2}$$

Considering E_b/N_0 and the available bandwidth (named B) is equal for both modulations, the only variable that changes is the bit rate (R_b). Therefore, equation 7.3 calculates the SNR gain in dB.

$$\frac{SNR(BPSK)}{SNR(QPSK)} = \frac{10log(\frac{R_b(BPSK)}{B})}{10log(\frac{2\times R_b(BPSK)}{B})} = 10log(2 \times \frac{R_b}{R_b}) = 10log(2) = 3dB \tag{7.3}$$

It's also noteworthy the negative effects of the ZF equalizer on an AWGN channel, jeopardizing it by 4dB approximately, whether BPSK or QPSK are used.

Furthermore, there is an improvement of approximately 2dB if turbo encoder is used. This statement is truth for both for BPSK and QPSK. This gain is obtained when the turbo decoder uses four iterations - sub-subsection 4.1.2.2. The gain value can increase or decrease with the number of iterations used.

BER - fading channel models: In this subsection, the BER is calculated for the three fading channel models provided by 3GPP, with added AWGN. Only QPSK modulation scheme is tested.

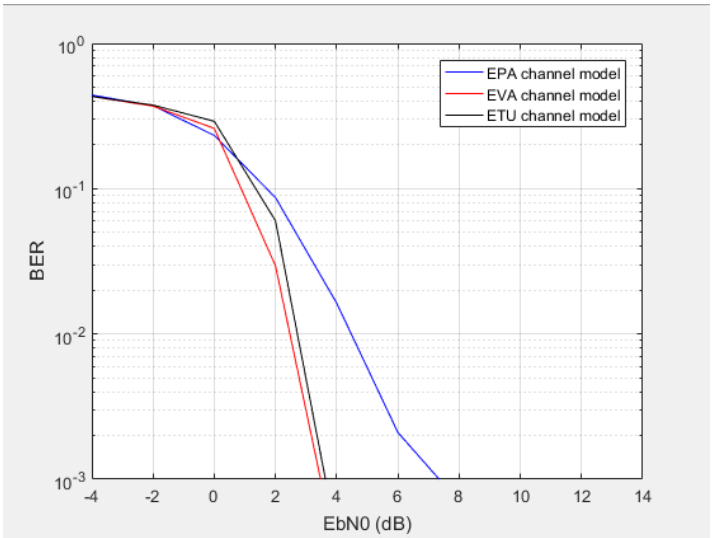


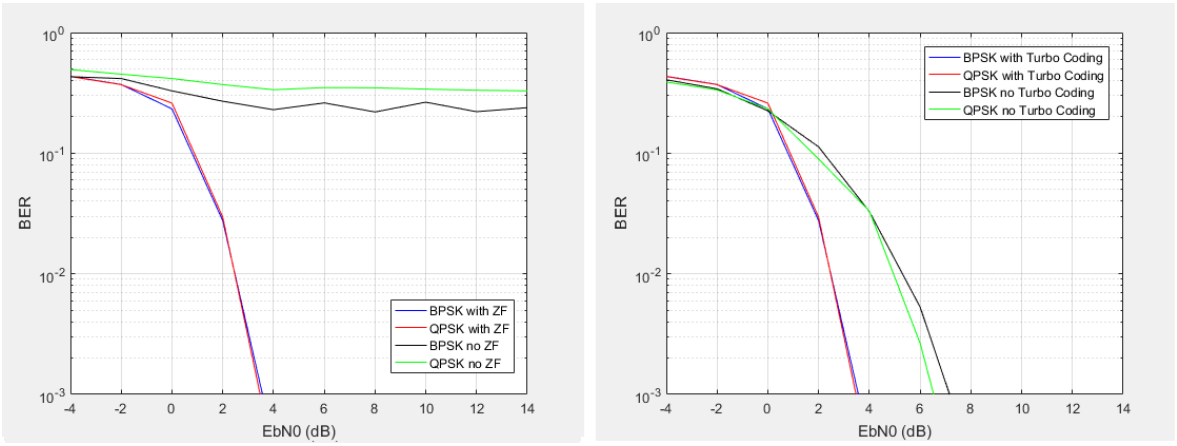
Figure 7.19: BER performance results for EPA, EVA and ETU channel fading models with added AWGN. Equalizer and turbo coding are used.

Observing Figure 7.19, it's possible to conclude the one causing more damage to the signal is the EPA channel model. The other two (EVA and ETU) show a similar BER and, therefore, disturb similarly the signal.

BER - fading channel with AWGN model: In this subsection, the BER is calculated considering the channel is composed by a channel fading component and AWGN. Performances with and without equalizer/turbo encoding are considered. QPSK and BPSK implementations are compared.

Figure 7.20a shows the BER for both QPSK and BPSK implementations with and without equalizer. Figure 7.20b displays the BER for both QPSK and BPSK implementations with

and without turbo encoder. Switches 1) and 2) of Figure 7.1 are used to connect/disconnect the turbo encoding and equalizer, respectively.



(a) BER performance results for an EVA channel model with AWGN, with and without equalizer for both QPSK and BPSK.

(b) BER performance results for an EVA channel model with AWGN, with and without turbo coding for both QPSK and BPSK.

Figure 7.20: BER performance results for a EVA channel model with AWGN.

Although BPSK and QPSK have the same BER performance, the SNR will differ by 3dB for the same BER. The reason behind this is demonstrated in equations 7.1, 7.2 and 7.3.

It's also noteworthy, the positive effects of the ZF equalizer on a fading channel. If the equalizer is not present, it is impossible to receive the signal accurately, independently of the E_b/N_0 value. This happens, whether BPSK or QPSK are used. Furthermore, there is an improvement of approximately 2dB if turbo encoder is used. This statement is true for both BPSK and QPSK.

Comparing QPSK and BPSK bandwidth efficiency, the higher order modulation (QPSK) accommodates more data within a given bandwidth and is more efficient when compared to lower order modulations.

Comparing this performance with the one where only AWGN is taken into account (Figure 7.18), they differ in about 2dB. This means that the EVA fading channel model, when ZF equalizer is used, jeopardizes the BER performance in about 2dB.

To conclude, BER is a key parameter for indicating the system performance.

Number of repetitions: In this subsection, it is tested whether the number of repetitions influences the BER. To avoid repeated results, only QPSK is considered at this stage. Values used for the signal generation are: $\Delta f = 15$kHz, RNTI $= 1$, $I_{sc} = 18$, $I_{MCS} = 2$, $I_{RU} = 2$ and $rv_{dci} = 0$. I_{sc} value is changed to provide some degree of comparison if, whether or not, the number of used subcarriers influence BER results. I_{Rep} parameter is variable between 0 and 7, and, therefore, N_{Rep} will vary from 1 to 128 - Table 5.4.

It's clear, observing Figure 7.21, the number of repetitions used influence BER results. When more repetitions are used, it's possible to obtain the same BER with a smaller E_b/N_0 present. With each N_{Rep} increase, a gain of approximately 2dB is observed. This is most noticeable when the first increases happen and less visible between higher numbers of N_{Rep}.

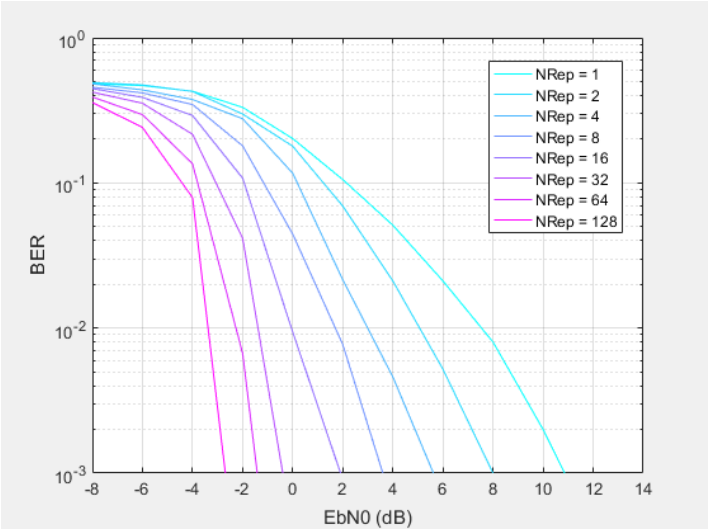
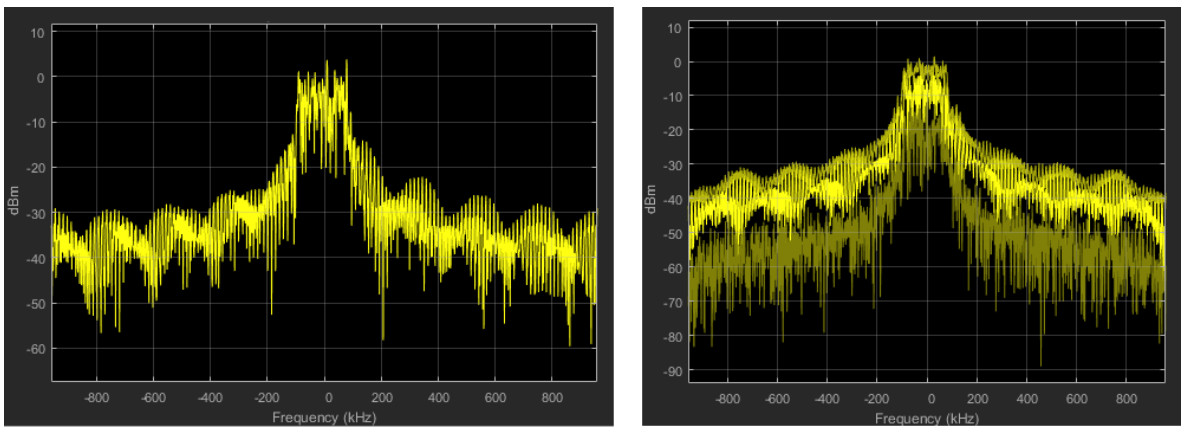


Figure 7.21: BER performance results for each N_{Rep} value.

Finally, when comparing with Figure 7.20, both have similar curves when I_{Rep} =3/N_{Rep} = 8, even though the I_{sc} value is different. Thus, the number of subcarriers is not related to the BER results.

7.1.1.6 Magnitude Spectrum

A spectrum analyzer measures the magnitude of an input signal in dBm versus frequency. Using the built-inMATLAB function `'spectrumAnalyzer'`, it is possible to obtain the transmitted and received (after channel effects) spectrums. Values used for the signal generation are: Δf = 15kHz, RNTI = 1, I_{sc} = 18, I_{Rep} = 3, I_{MCS} = 2, I_{RU} = 2 and rv_{dci} = 0.



(a) Transmitted magnitude spectrum.

(b) Received magnitude spectrum.

Figure 7.22: Transmitted and received magnitude spectrums.

On Figure 7.22, it is clear the channel effects that alter the transmitted spectrum (Figure 7.22a) into the received one (Figure 7.22b). Results are according to expected, since the

bandwidth is of approximately 180kHz - section 2.3. The signal magnitude is around 0dBm.

7.1.1.7 PAPR

Power saving in transmission is an important issue in IoT applications. Therefore, PAPR is an important factor to be considered.

PAPR is calculated by representing a Complementary Cumulative Distribution Function (CCDF). The CCDF is the probability of the PAPR being higher than a specific value (PAPR0). It is calculated using the MATLAB built-in functions `'hist'` and `'cumsum'`.

QPSK and BPSK results are similar and, therefore, only the QPSK ones are displayed.

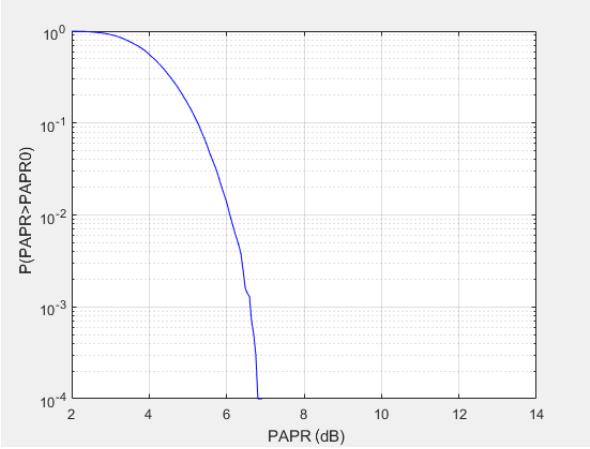


Figure 7.23: SC-FDMA PAPR.

PAPR has a reasonably low value, which is according to expected. If compared with OFDMA values presented on [SMM13], it has a lower value, which is ideal for power saving on uplink transmissions.

7.1.2 NPUSCH Format 2 Simulation Results

This subsection displays the results regarding the NPUSCH format 2. Figure 7.24 presents a schematic of the MATLAB implemented chain. It is divided in coding/decoding procedures, which is in blue color and modulation/demodulation procedures which is in green color. Points where constellations and eye diagrams were obtained are represented by the letter a) and b). The first corresponds to the results on the transmitter side and the second on the receiver side. Several switches are represented throughout the figure, using points 1) and 2). 1) is used to measure BER results. Essentially, that part was bypassed on the code, so BER gains could be measured with and without that specific step. This way the gain difference could be obtained and, as a consequence, their importance in the general chain could be assessed. Point 1) corresponds to the equalizer bypass. Point 2) is in Figure 7.24 to represent the several channels that can be tested. Each `mode` corresponds to a different channel model. More information about each mode can be found on sub-subsection 6.2.2.2. Finally, in the blue circles are the transmitted and received transport blocks. When compared, the BER performance can be calculated. In appendix B.1, a known sequence was imposed on the beginning of the chain so a reference sequence test is available in each main point of the implemented simulation.

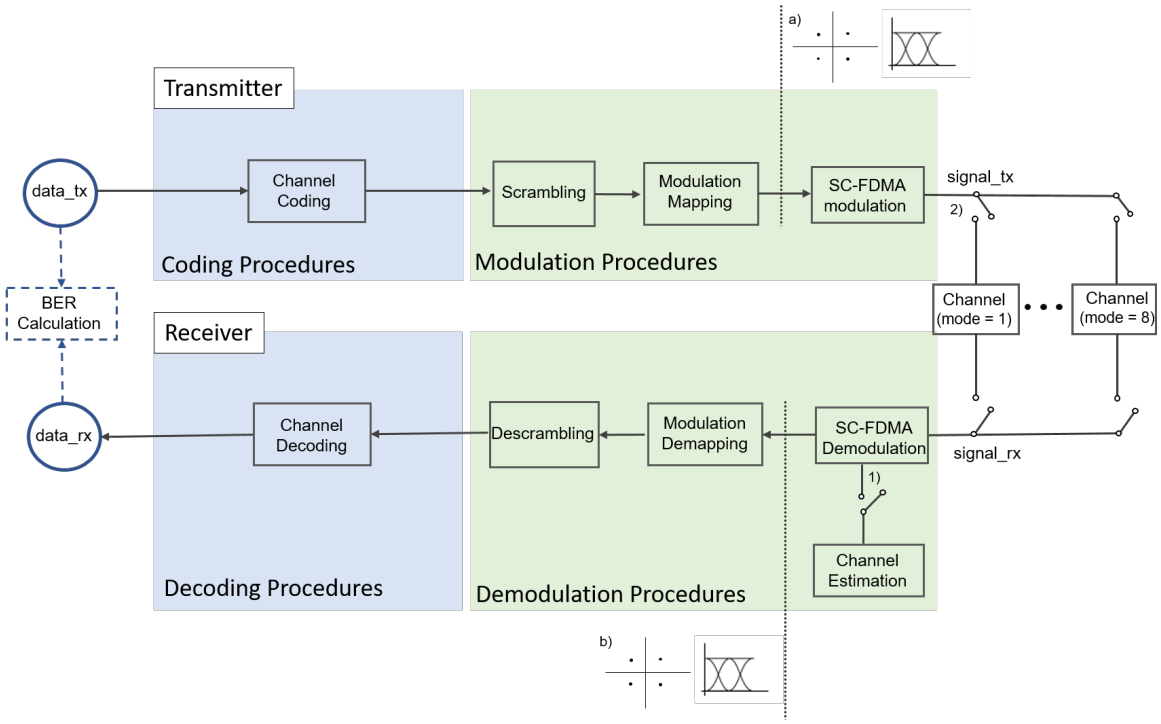


Figure 7.24: MATLAB implemented chain schematic for NPUSCH format 2. Measures of constellations and eye diagrams happen on points a) and b).

Several measures are taken and analyzed throughout this subsection. A small overview of them is made below:

1. On sub-section 7.1.2.1, it is shown the transmitted constellation for BPSK - only modulation scheme used on NPUSCH format 2. These results are obtained on point a) of Figure 7.24.

2. On sub-section 7.1.2.2, it is presented the received constellations, when using 3GPP specified fading channel models with added AWGN. Only the EVA model was tested - on point 2) of Figure 7.24, `mode` = 6. Only the BPSK modulation scheme was available to be evaluated. Two different E_b/N_0 values were analyzed - 8dB and 12dB. These results are obtained on point b) of Figure 7.24.

3. On sub-section 7.1.2.3, it is analyzed the BER performances, when using the EVA channel model with AWGN - on point 2) of Figure 7.24, `mode` = 6. Tests with and without equalizer (switch represented by point 1) were made.

4. On sub-section 7.1.2.4, it is provided a small explanation why no other results were measured.

Values used when coding and modulating the transport block are: $\Delta f = 15$kHz, $N_{Rep}^{AN} = 4$, ACK/NACK resource field = 5 and RNTI = 1. These values are constant throughout this subsection, unless otherwise mentioned.

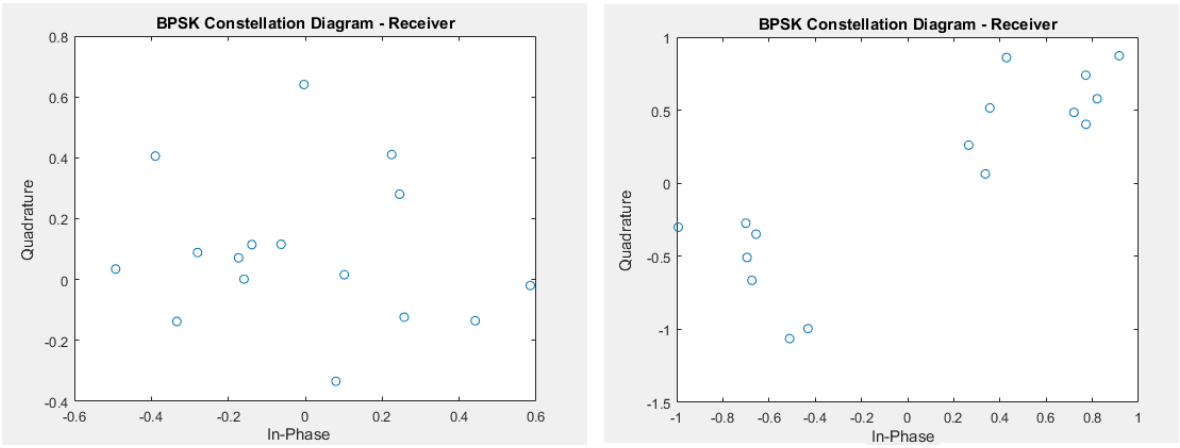
7.1.2.1 Constellation - Transmitter

The transmitted constellation is equal to the ones displayed on subsection 7.1.1.1, for BPSK (only modulation scheme available on NPUSCH format 2).

7.1.2.2 Fading Channel with AWGN - Constellations

This sub-subsection considers the channel is composed by a fading channel model (EVA) with AWGN - on point 2) of Figure 7.24, `mode` = 6. Two values of E_b/N_0 were tested.

First, the constellation for E_b/N_0 equal to 8dB is depicted in Figure 7.25. Figure 7.25a shows the constellation without equalizer and Figure 7.25b depicts the constellation with equalizer. Secondly, results for E_b/N_0 equal to 12dB are also depicted. Figure 7.26a shows the constellation without equalizer and Figure 7.26b presents the constellation with equalizer. No eye diagram results are shown.



(a) BPSK constellation for EVA channel model with AWGN, when no equalizer is used.

(b) BPSK constellation for EVA channel model with AWGN, when ZF equalizer is used.

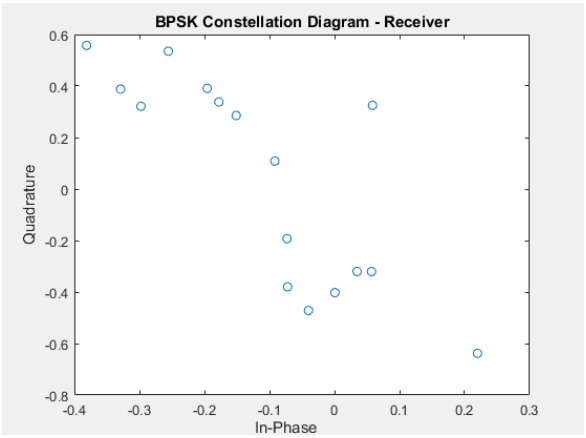
Figure 7.25: BPSK constellation for EVA channel model with AWGN (E_b/N_0 is 8dB).

It's possible to conclude the results with equalizer are considerably better, which is expected. Both signals are possible to recuperate. For lower values of E_b/N_0 where more AWGN is present the signal won't be possible to decode. This means an equalizer with better performance (which takes AWGN into consideration), should be utilized in such situations.

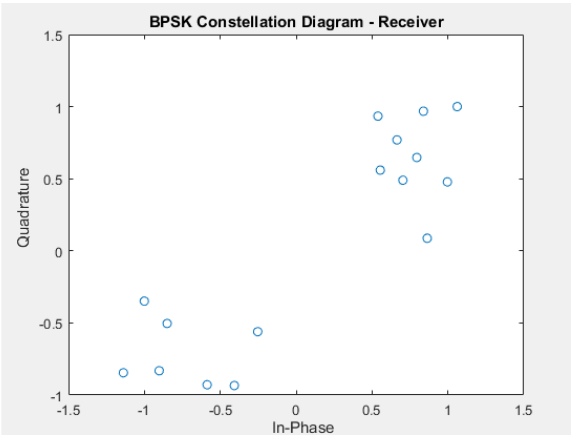
To conclude, the signal is always recovered, as long as ZF equalizer is used and E_b/N_0 isn't lower then 8dB. This value is obviously variable, depending on the number of repetitions used as explained in subsection 7.1.1.5.

7.1.2.3 BER

In all simulations, BER tests were run 1000 times for each E_b/N_0. Performance was evaluated considering the channel is composed by a channel fading component (EVA model) and AWGN. Performances with and without equalizer are considered - point 1) on Figure 7.24. Figure 7.27 shows the BER with and without equalizer.



(a) BPSK constellation for EVA channel model with AWGN, when no equalizer is used.



(b) BPSK constellation for EVA channel model with AWGN, when ZF equalizer is used.

Figure 7.26: BPSK constellation for EVA channel model with AWGN (E_b/N_0 is 12dB).

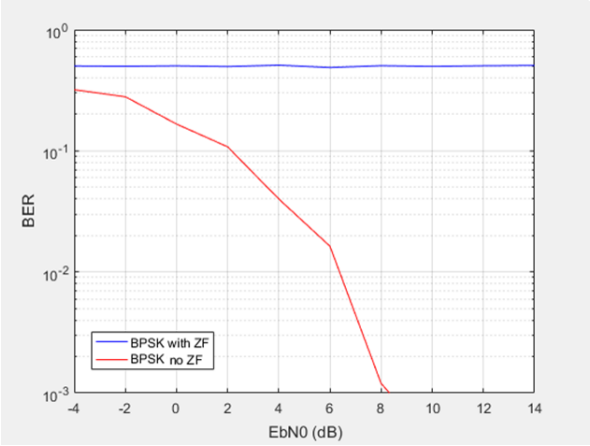


Figure 7.27: BER performance results for a EVA channel model with AWGN, with and without equalizer.

It is noteworthy the positive effects of the ZF equalizer on the fading channels. If the equalizer is not present, it is impossible to accurately receive the signal, independently of the E_b/N_0 value.

When comparing with NPUSCH format 1's performance, format 2 is considerably worst. This is due to the fact that no turbo coding is used in format 2. Furthermore, the number of repetitions used corresponds to half the ones used on the format 1 testing. Therefore, results are according to expected.

7.1.2.4 Repetitions Test, PAPR and Magnitude Spectrum

Since the results are similar to the ones for NPUSCH format 1, these performance tests won't be displayed.

7.1.3 NPRACH Simulation Results

This subsection displays the results regarding the NPRACH. Figure 7.28 presents a schematic of the implemented chain. It is divided in transmitter and receiver parts. Points where constellations and eye diagrams were obtained are represented by the letter a) and b). The first corresponds to the results on the transmitter side and the second on the receiver side. Point 1) is in Figure 7.28 to represent the several channels that can be tested. Each `mode` corresponds to a different channel model. More information about each mode can be found on sub-subsection 6.2.2.2.

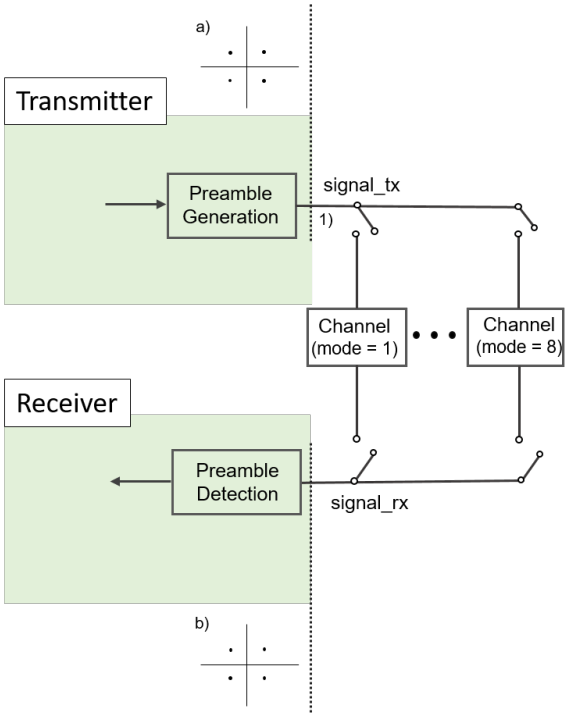


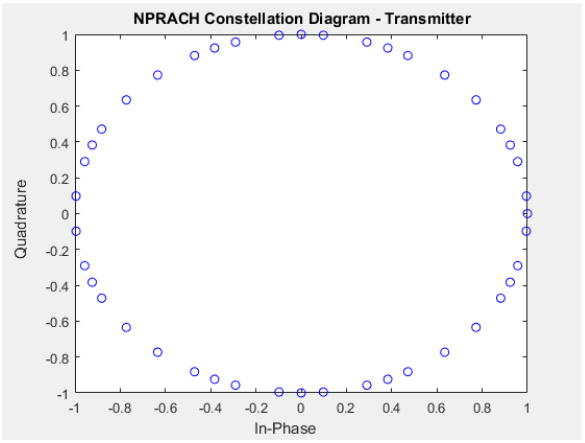
Figure 7.28: MATLAB implemented chain schematic for NPRACH. Measures of constellations and cross-correlations happen on points a) and b).

Several measures are taken and analyzed throughout this subsection. A small overview of them is made below:

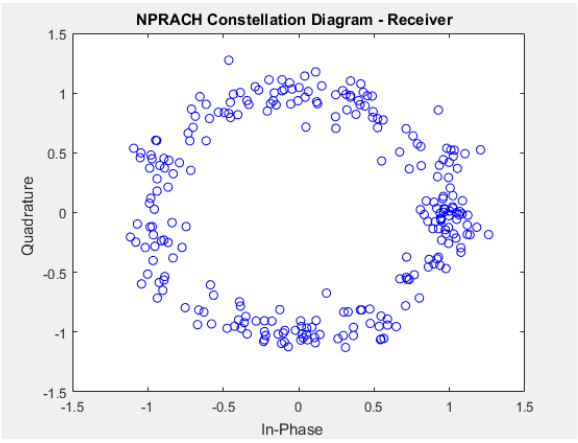
1. On sub-section 7.1.3.1, it is displayed the transmitted and received constellations. These results are obtained on point a) and b) of Figure 7.24, respectively.
2. On sub-section 7.1.3.2, it is presented the transmitted preamble auto-correlation, measured on point a) of Figure 7.24. The cross-correlation between the expected preamble and the received one, measured on point b) of Figure 7.24, is also displayed.

7.1.3.1 Constellation

Figure 7.29 displays the NPRACH transmitted and received constellations. An AWGN channel model was used (on Figure 7.28, on point 1), `mode` = 1), with E_b/N_0 equal to 4dB. To run this simulation the values used were: $N_{scoffset}^{NPRACH} = 0$, $N_{sc}^{NPRACH} = 12$, $N_{rep}^{NPRACH} = 1$ and NPRACHFormat = 1.



(a) NPRACH transmitted constellations.



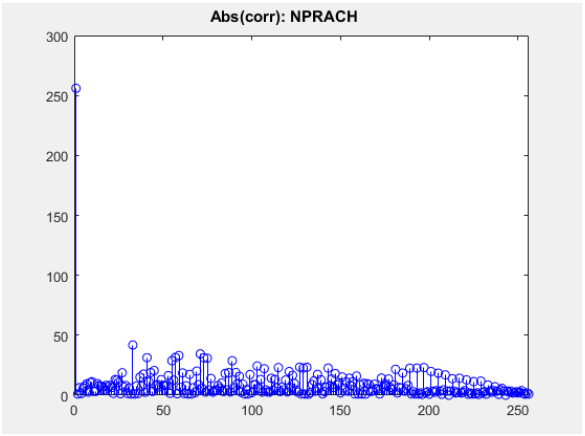
(b) NPRACH received constellations.

Figure 7.29: NPRACH transmitted and received constellations.

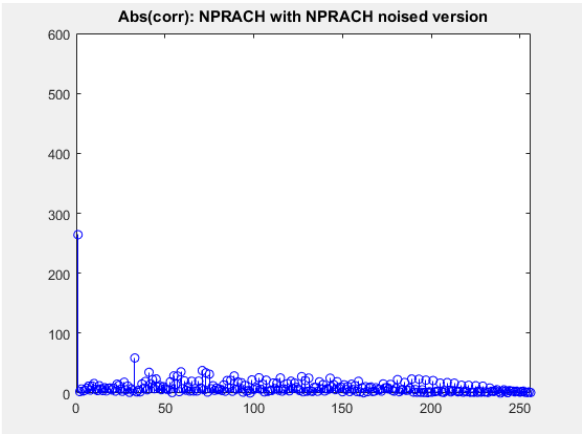
On Figure 7.29a is noticeable the preamble is a ZC sequence due to its unique circular constant amplitude constellation. On Figure 7.29b, the typical circular format is slightly altered due to the added AWGN.

7.1.3.2 Correlation

Two identical ZC sequences have the property of their cross-correlation being very high when the lag between both is zero. When there is any time discrepancy, the correlation is a low value near zero. Figure 7.30a shows the auto-correlation of the transmitted signal and Figure 7.30b displays the cross-correlation between the expected and the received preamble.



(a) Auto-correlation of the transmitted preamble.



(b) Cross-correlation between the expected and received preamble.

Figure 7.30: Auto-correlation of the transmitted preamble and cross-correlation between the expected and received preamble.

Figure 7.30a clearly illustrates the ZC sequence auto-correlation property. Although AWGN is added on Figure 7.30b, it is still clear the ZC cross-correlation properties.

7.2 USRP Co-simulation Results

Implementation results for NPUSCH format 1 and NPUSCH format 2 co-simulation are displayed in this section. Magnitude spectrums, constellations, eye diagrams and BER performances are measured. Appendix A is a small user guide that exemplifies how to run all the possible simulations and co-simulation. Section A.4 describes how to run the co-simulation using two computers and two USRPs.

Figure 7.31 presents a schematic of the co-simulation implemented chain. It is divided in coding/decoding procedures, which is in blue color, modulation/demodulation procedures which is in green color and the RF front-end part which is in pink color. Points where constellations and eye diagrams were obtained are represented by the letter a) and b). The first corresponds to the results on the transmitter side and the second on the receiver side. Furthermore, in the blue circles are the transmitted and received transport blocks. When both are compared, the BER performance can be measured.

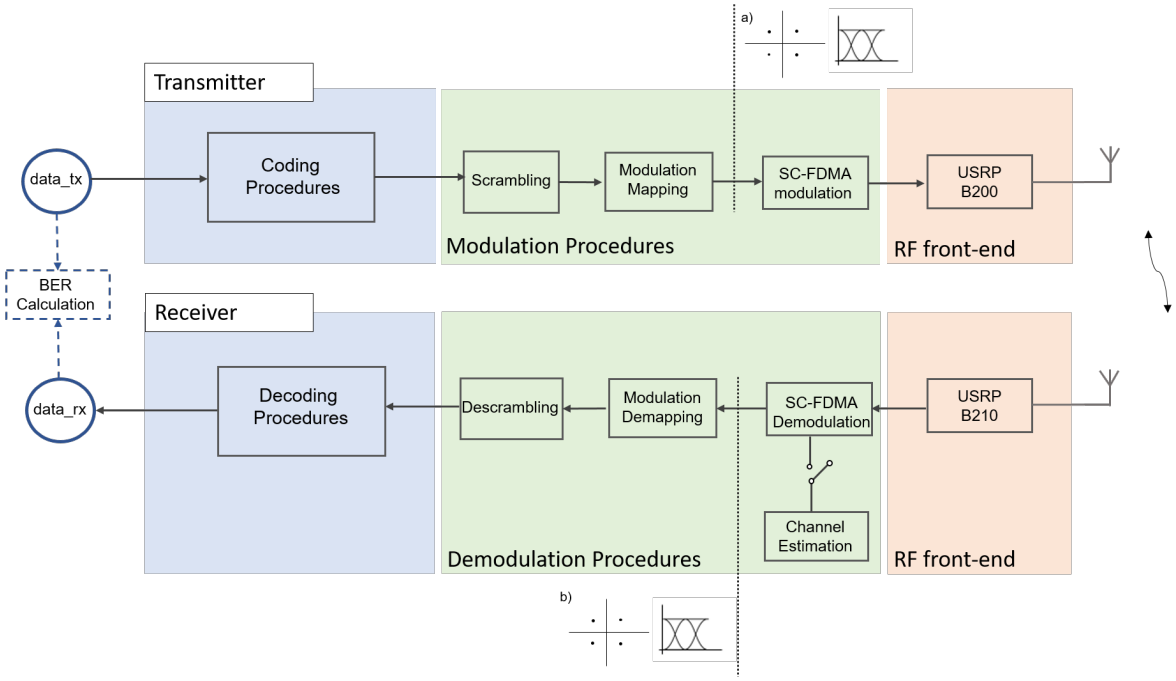


Figure 7.31: Implemented chain schematic for a co-simulation environment. Measures of constellations and eye diagrams happen on points a) and b).

Several measures are taken and analyzed throughout this section. A small overview of them is made below:

1. On subsection 7.2.1, NPUSCH format 1 is analyzed. This section is divided in 4 sub-subsections. Those are:
    - On sub-subsection 7.2.1.1, it is displayed the transmitted constellation and eye diagram, for both BPSK and QPSK modulation schemes. These results are obtained

on point a) of Figure 7.31.

- On sub-subsection 7.2.1.2, it is depicted the received constellation and eye diagram, for both BPSK and QPSK modulation schemes. Thse results are obtained on point b) of Figure 7.31.
- On sub-subsection 7.2.1.3, it is analyzed the BER performances, obtained using a known transmitted sequence. Performance is evaluated on how BER varies with the distance between USRPs, and how BER varies with N_{Rep}.
- On sub-subsection 7.2.1.4, it is presented transmitted and received magnitude spectrums.

2. On subsection 7.2.2, NPUSCH format 2 is analyzed. This subsection has only one sub-subsection, to avoid showing similar results. On sub-subsection 7.2.2.1, it is shown the receiver constellation for BPSK - only modulation scheme available on NPUSCH format 2. These results are obtained on point b) of Figure 7.31.

Values used when coding and modulating the transport block are: $\Delta f = 15$kHz, RNTI = 1, $I_{sc} = 18$, $I_{MCS} = 2$, $I_{RU} = 2$, $I_{Rep} = 3$ and $rv_{dci} = 0$ for QPSK and $\Delta f = 15$kHz, RNTI = 1, $I_{sc} = 5$, $I_{MCS} = 0$, $I_{RU} = 2$, $I_{Rep} = 3$ and $rv_{dci} = 0$ for BPSK. These values are constant throughout this subsection, unless otherwise mentioned.

7.2.1 NPUSCH Format 1

This subsection displays results regarding the NPUSCH format 1 co-simulation.

7.2.1.1 Constellation and Eye Diagram - Transmitter

Constellation and eye diagram are equal to the ones displayed on subsection 7.1.1.1, for both BPSK and QPSK.

7.2.1.2 Constellation and Eye Diagram - Receiver

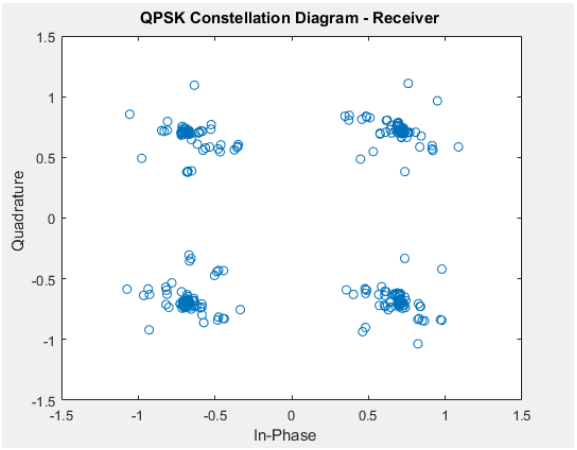
Figure 7.32 shows the received constellations. Results are displayed for QPSK on Figure 7.32a and for BPSK on Figure 7.32b. Figure 7.33 shows the transmitted eye diagram. Results are displayed for QPSK on Figure 7.33a and BPSK on Figure 7.33b.

When comparing the transmitted and received constellations, even though channel effects are clear in the received ones, both are similar. Thus, the typical two/four points focuses are visible, for BPSK and QPSK, respectively. Regarding the eye diagram, the middle eye is visible and wide open, which is ideal.

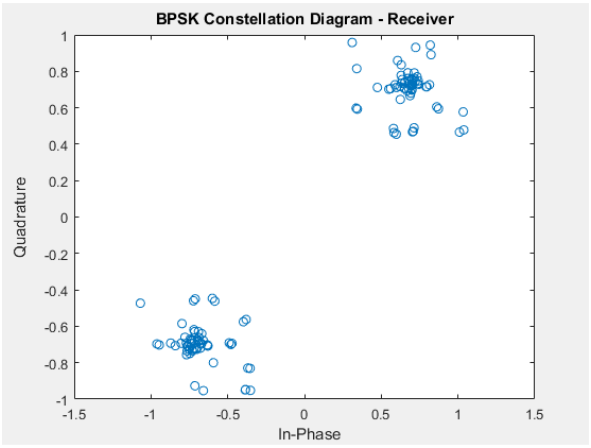
7.2.1.3 BER

In all simulations, BER tests were run 1000 times, with each result being averaged afterwards. The performance could be evaluated, since all the transmitted transport blocks were known at the receiver, so the number of bit errors could be accounted for.

Table 7.1 shows the calculated BERs, depending on the distance between both USRPs. Table 7.2 presents the obtained BERs, depending on N_{Rep}.

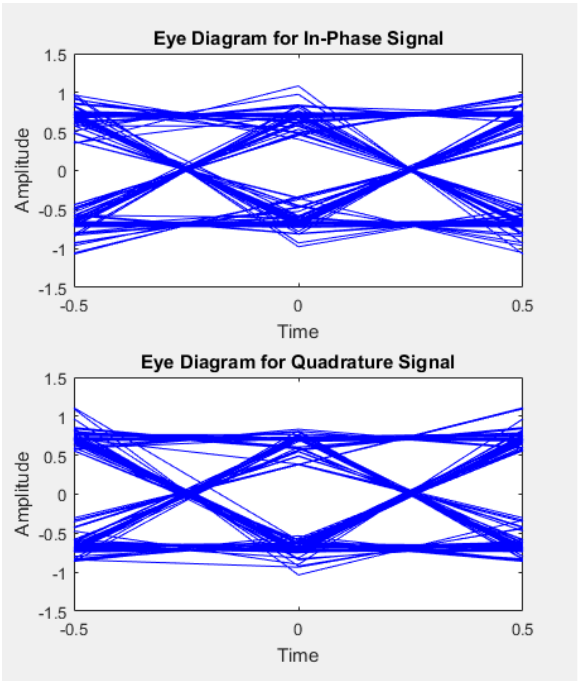


(a) QPSK received constellation.

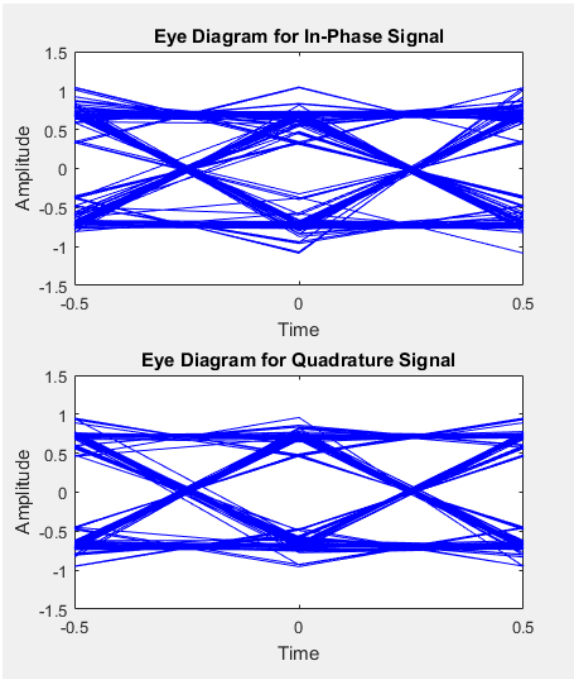


(b) BPSK received constellation.

Figure 7.32: QPSK and BPSK received constellations for NPUSCH format 1.



(a) QPSK received eye diagram.



(b) BPSK received eye diagram.

Figure 7.33: QPSK and BPSK received eye diagrams for NPUSCH format 1.

Table 7.1: Variation of the co-simulation BER with the distance between USRPs.

Distance(cm)	1	10	100	200
BER	0.0475	0.0557	0.0644	0.0689

Table 7.2: Variation of the co-simulation BER with I_{Rep}.

N_{Rep}	1	2	4	8	16	32	64	128
BER	0.1043	0.0840	0.0670	0.0541	0.0475	0.0451	0.0420	0.0390

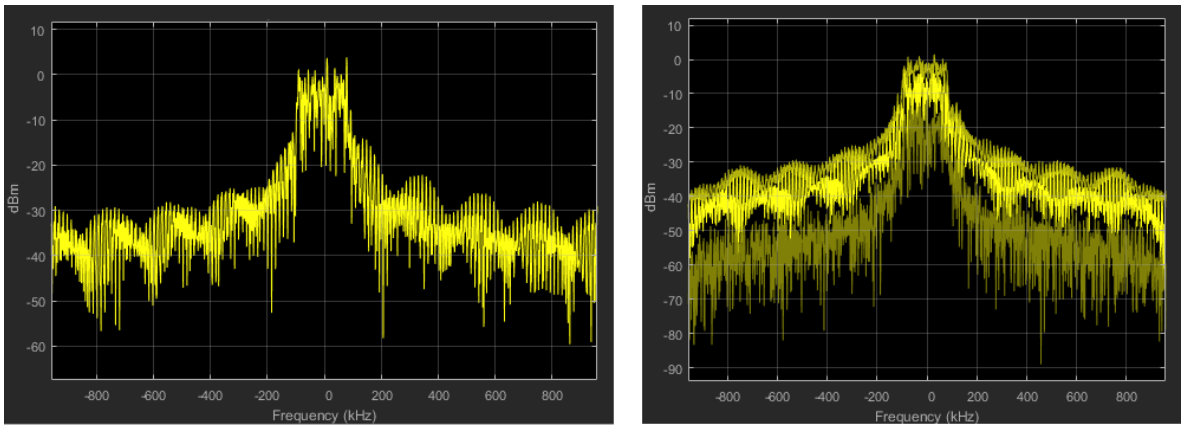
It is clear that the BER increases with distance, which is expected. Also, it's possible to conclude the BER decreases with I_{Rep}, which is, again, predictable.

Taking into account how close to each other the USRPs are, BER values are quite high, specially when comparing with simulation results. Since CFO and STO synchronization functions were not tested in the simulation environment and clearly have room for improvement, they should be responsible for these values. Upgrading their algorithms can improve the obtained results.

It still makes sense that the BER decreases with repetitions, since more DMRS are available when performing correlation in the STO step and more SC-FDMA symbols are obtained, which helps with the CFO synchronization.

7.2.1.4 Magnitude Spectrum

Using the built-in MATLAB function `'spectrumAnalyzer'`, it is possible to obtain the magnitude spectrum of the transmitted and received baseband signals. On Figure 7.34, it is clear the channel effects that alter the transmitted spectrum (Figure 7.34a) into the received one (Figure 7.34b). Results are according to expected, since the bandwidth is of approximately 180kHz - section 2.3. The signal magnitude is around 0dBm.



(a) Magnitude spectrum of the transmitted signal.

(b) Magnitude spectrum of the received signal.

Figure 7.34: Magnitude spectrum of the transmitted and received signals.

7.2.2 NPUSCH Format 2

This subsection displays the results regarding NPUSCH format 2. Eye diagrams and BERs are not displayed, since the results are similar to the ones for NPUSCH format 1.

7.2.2.1 Constellation - Receiver

Figure 7.35 shows the received constellation. Only BPSK modulation scheme is used on NPUSCH format 2.

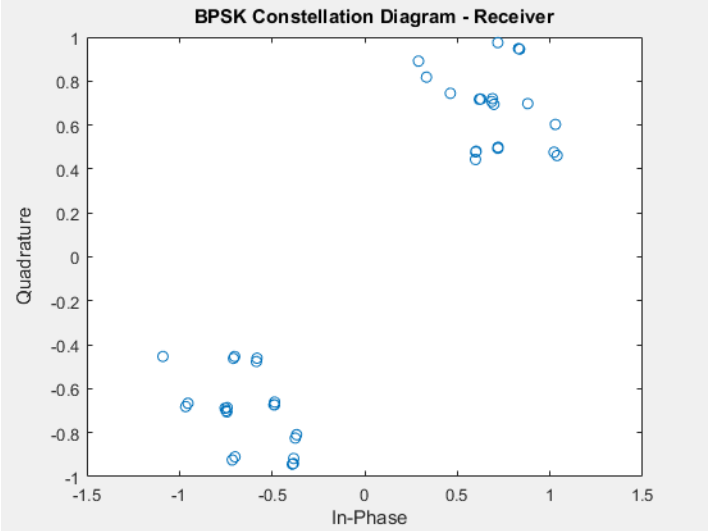


Figure 7.35: BPSK received constellation for NPUSCH format 2.

The constellation is according to expected, with two focus of points clearly representing a BPSK signal.

Throughout chapter 6, both the MATLAB simulation and the USRP co-simulation were described with the aid of block diagrams. In this chapter, results for both implementations were displayed, including eye diagrams, constellations, BERs, magnitude spectrums and PAPR values. With the end of the practical work description, a conclusion of what was developed will be provided on the next chapter. Besides providing a summary of the implemented work, it will present details that could be improved, regarding the final implementation.

Chapter 8

Conclusions and Future Work

This chapter is divided in two subsections: conclusions and future work. On the first one, a summary of the developed work is done. On the last one, some aspects that could be improved are presented, regarding both the MATLAB implementation and the USRP co-simulation.

8.1 Conclusions

Throughout this master thesis, an open-source uplink behavioral simulator based on MATLAB, mainly focused on the physical layer relevant functionalities, was implemented. Furthermore, a co-simulation environment was tested, where the MATLAB signal was transmitted and received using RF front-ends, consisting of two USRPs.

This document began by explaining some essential concepts to contextualize the final work. Transmitter and receiver architectures were described in detail and so were all the used channels and signals. Next, the necessary procedures to send/receive transport blocks in all channels were clarified, including all the parameters required to simulate each transmission/reception and their respective function.

Afterwards, the MATLAB simulation block diagram and its respective functions and work flow was illustrated. Furthermore, the USRP co-simulation addition was explained in detail.

Finally, all the results are depicted with their respective conclusions, including constellations, BERs, eye diagrams, PAPR analysis and magnitude spectrums.

8.2 Future Work

Even though the implementation gives a good starting point for the test and analysis of the uplink NB-IoT physical layer, there are some improvements that could be made.

A noteworthy improvement could be the introduction of a Minimum Mean-Squared Error (MMSE) equalizer, as it includes an AWGN estimation term, which would minimize the additive noise effects. The channel is currently estimated using a ZF equalizer only.

Further work could also be done to create a full virtual network, where this dissertation's implementation would be combined with the correspondent downlink implementation. Furthermore, implementations of upper layers could be added, both in the transmitter and receiver sides.

Finally, the CFO and STO estimation could be upgraded, using more sophisticated algorithms in their calculations. This would improve the BER performance, when using an over-the-air implementation.

Bibliography

[3GP16] 3GPP TS 36.211. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation, November 2016.

[3GP17a] 3GPP TS 36.212. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding, January 2017.

[3GP17b] 3GPP TS 36.213. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures, March 2017.

[Blo17a] Countableset Blog. *Cyclic Redundancy Check (CRC) Calculation*, 2017. [Online] Available at: `https://communities.theiet.org/blogs/426/424` [Accessed in April 29th, 2017].

[Blo17b] Countableset Blog. *SISO (single input, single output)*, 2017. [Online] Available at: `http://searchmobilecomputing.techtarget.com/definition/SISO` [Accessed in September 9th, 2017].

[CVZZ16] Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios. *IEEE Wireless Communications*, 23(5):60–67, 2016.

[DPS16] Erik Dahlman, Stefan Parkvall, and Johan Skold. *4G, LTE-Advanced Pro and The Road to 5G*. Academic Press, 2016.

[DT17] Piotr Krasowski Douglas Troha. NB-IoT downlink simulator. Master's thesis, Uppsala University, Uppsala, Sweden, February 2017.

[Gil03] James E Gilley. Bit-error-rate simulation using matlab. *Transcrypt International, Inc*, pages 1–7, 2003.

[GZAM10] Arunabha Ghosh, Jun Zhang, Jeffrey G Andrews, and Rias Muhamed. *Fundamentals of LTE*. Pearson Education, 2010.

[Imr13] Anum Imran. Software implementation and performance of umts turbo code. 2013.

[JKD16] Aanchal Jhingan, Lavish Kansal, and Navdeep Singh Dhaliwal. Comparative study of diverse cfo estimation techniques in ofdm system. *Indian Journal of Science and Technology*, 9(47), 2016.

[Joo10] Michael Joost. Turbo codes. 2010.

[LAW16] Xingqin Lin, Ansuman Adhikary, and Y-P Eric Wang. Random access preamble design and detection for 3gpp narrowband iot systems. *IEEE Wireless Communications Letters*, 5(6):640–643, 2016.

[LGV14] Ana Rita Luzio, João Venturinha Gomes, and Pedro Vieira. Performance Gain Evaluation from High Speed Packet Access Evolution (HSPA+). *Procedia Technology*, 17:720–727, 2014.

[Li09] Liang Li. Analysis of low power implementational issues of turbo-like codes in body area networks. *Progress report for continuation towards PhD, University of Southampton*, 2009.

[Mat] Mathworks. *Channel Estimation.* [Online] Available at: `https://www.mathworks.com/help/lte/ug/channel-estimation.html` [Accessed in September 12th, 2017].

[Mat17a] MathWorks. *SDRu Receiver*, 2017. [Online] Available at: `https://www.mathworks.com/help/supportpkg/usrpradio/ug/sdrureceiver.html` [Accessed in May 12th, 2017].

[Mat17b] MathWorks. *SDRu Transmitter*, 2017. [Online] Available at: `https://www.mathworks.com/help/supportpkg/usrpradio/ug/sdrutransmitter.html` [Accessed in May 12th, 2017].

[Mat17c] MathWorks. *Uplink Shared Channel*, 2017. [Online] Available at: `https://www.mathworks.com/help/lte/ug/uplink-shared-channel.html` [Accessed in May 2nd, 2017].

[Mau10] Rob Maunder. *Matlab UMTS/LTE Turbo Code*, 2010. [Online] Available at: `http://users.ecs.soton.ac.uk/rm/resources/matlabturbo/` [Accessed in February 12th, 2017].

[Not09] Agilent Application Note. 3gpp long term evolution: System overview, product development, and test challenges. *Literature Number*, 2009.

[oE17] USC Viterbi School of Engineering. *Software Defined Radio*, 2017. [Online] Available at: `http://cci.usc.edu/wp-content/uploads/2017/09/CLASS-3-SOFTWARE-TOOLS.pdf` [Accessed in May 12th, 2017].

[RBB17] Kamisetty Ramamohan Rao, Zoran S Bojkovic, and Bojan M Bakmaz. *Wireless multimedia communication systems: design, analysis, and implementation.* CRC Press, 2017.

[Roh16] Rohde & Schwarz. *Narrowband Internet of Things Whitepaper.* August 2016.

[SMM13] Bale Mallikarjun Sidram, TP Mithun, and M Madhukar. Ber, snr, papr analysis for multiple accesses in lte. *Int. J. of Scientific & Engineering Research*, 4(7):773–7, 2013.

[SP16] Albert Serra Pagès. Link level performance evaluation and link abstraction for lte/lte-advanced downlink. 2016.

[SWH17] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. A survey on lpwa technology: Lora and nb-iot. *ICT Express*, 2017.

[Tel15] Teletopix. *Modulation Techniques In LTE*, 2015. [Online] Available at: `http://www.teletopix.org/4g-lte/modulation-techniques-in-lte/` [Accessed in May 11th, 2017].

[YPEWR17] Ansuman Adhikary Asbjrn Grvlen Yutao Sui Yufei Blankenship Johan Bergman Y.-P. Eric Wang, Xingqin Lin and Hazhir S. Razaghi. A Primer on 3GPP Narrowband Internet of Things (NB-IoT). *IEEE Communications Magazine*, 55(3):117–123, March 2017.

Appendix A

User Guide

This annex provides one guide on how to run the NPUSCH format 1/2 simulation (section A.1), one on how to run the NPRACH simulation (section A.2) and one on how to calculate the BER (section A.3). Furthermore, an explanation on how to perform co-simulation using two USRPs is also available (section A.4).

A.1 NPUSCH Format 1/2 Simulation

This section presents a small user guide, explaining how to run the NPUSCH format 1 and 2 simulation. The several steps are explained below:

- First, open `RunDataSim()` MATLAB program. On top of the program, there is a selection of parameters that can be changed by the user. If no value is added, when the parameter is needed for calculations, one of the possible values is automatically assigned. The list of parameters that can be changed by the user, is presented on Figure A.1.

```
%%%%%%%%%% Select Format %%%%%%%%%%%%%%%
Format = [];

%%%%%%%%%% If Format = 1 %%%%%%%%%%%%%%%
deltaf = [];
Isc = [];
Irep = [];
Iru = [];
Imcs = [];
RVdci = [];
RNTI = [];
goup_hopping = [];

%%%%%%%%%% If Format = 2 %%%%%%%%%%%%%%%
deltaf = [];
ACKNACK = [];
NANRep =[];
RNTI = [];

%%%%%%%%%% Channel Variables %%%%%%%%%%
EbN0 = [];
mode = [];
```

Figure A.1: User input parameters list for NPUSCH format 1 and 2 simulation.

- Then, run the `RunDataSim()` program. This will run the transmitter, channel and receiver explained on section 6.2. When running this simulation, transmitted and received constellations/eye diagrams are shown. If NPUSCH format 1 is selected, the text presented in Figure A.2 appears on the command window.

```
--------------------------- Transmitter ----------------------------
Transmitted data: 1 0 1 1 1 0 1 1 0 0 1 0 0 0 1 0
---------------------------  Receiver  -----------------------------
Received repeated data streams:
1011101100100010
1011101100100010
1011001000100010
1011101100100010
1011101100100010
1011101100100010
1111100111110101
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
Vector with number of turbo coding iterations used to recover each stream (max value: 4):
2 2 4 2 2 2 4 2 2 2 2 2 2 2 2 2
Positions of incorrectly recovered data streams:
0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
Recovered data streams after incorrect vectors are removed:
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
1011101100100010
Recovered data after bit combining: 1 0 1 1 1 0 1 1 0 0 1 0 0 0 1 0
Number of slots: 32
Number of received subframes: 16
Number of received frames: 1
```

Figure A.2: Command window output for NPUSCH format 1.

First, the transmitted transport block is displayed. On the receiver side, it is shown all the recovered transport blocks (one for each repetition), even if the CRC check part showed errors were present. For each repeated data stream, it is displayed how many iterations were used in the turbo decoder phase and if errors were detected in the CRC

check step. After, it's presented only the transport blocks where no errors were detected. Finally, the recovered transport block is depicted, after all repetitions are combined. To conclude, the transmitted data stream is equal to the received one. The number of slots, subframes and frames utilized is also shown.

If NPUSCH format 2 is chosen, the text depicted in Figure A.3 is displayed on the command window. First, the transmitted codeword is shown. On the receiver side, it is printed all the recovered codewords (one for each repetition). Finally, the recovered codeword is depicted, after all repetitions are combined. To conclude, the transmitted data stream is equal to the received one. The number of slots, subframes and frames utilized is also shown.

```
------------------- Transmitter -----------------
Transmitted codeword: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

-------------------- Receiver -------------------
Received repeated codewords:
0000000000000000
0000000000000000
0010000000000000
0000000000000000

Recovered codeword: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Number of slots: 16
Number of received subframes: 8
Number of received frames: 0
```

Figure A.3: Command window output for NPUSCH format 2.

A.2 NPRACH Simulation

This section presents a small user guide, explaining how to run the NPRACH simulation. The several steps are explained below:

- First, open `RunNprachSim()` MATLAB program. On top of the program, there is a selection of parameters that can be changed by the user. If no value is added, when the parameter is needed for calculations, one of the possible values is automatically assigned. The list of parameters that can be changed by the user, is presented on Figure A.4.
- Then, run the `RunNprachSim()` program. This will run the transmitter, channel and receiver explained on section 6.2. When running this simulation, transmitted and received constellations are shown. Also, plots with auto-correlation and cross-correlation values appear automatically. On the command window, it is written if the NPRACH preamble is detected or not.

```
%%%%%%%%%% Channel Variables %%%%%%%%%%
EbN0 = [];
mode = [];

%%%%%%%%%% Preamble Variables %%%%%%%%%%
NPRACHFormat = [];
NNPRACHsc = [];
NNPRACHscoffset = [];
NNPRACHrep = [];
NCELLid = 1;
```

Figure A.4: User input parameters list for NPRACH simulation.

A.3 BER Performance Simulation

This section presents a small user guide, explaining how to calculate the NPUSCH format 1 and 2 BERs. The several steps are explained below:

- First, open `RunBerSim()` MATLAB program. On top of the program, there is a selection of parameters that can be changed by the user. That list is presented on Figure A.1.

```
%%%%%%%%%% If Format = 1 %%%%%%%%%%%%%%
deltaf = [];
Isc = [];
IRep = [];
IRu = [];
Imcs = [];
rvdci = [];
RNTI = [];
GH = [];

%%%%%%%%%% If Format = 2 %%%%%%%%%%%%%%
deltaf = [];
ACKNACK = [];
NANRep = [];
RNTI = [];

%%%%%%%%%% Channel Variables %%%%%%%%%%
mode = [];

%%%%%%%%%% Run BER Parameters %%%%%%%%%
N = 500; % Number of simulations
EbN0 = -16:2:0; % EbN0 interval to be tested
```

Figure A.5: User input parameters list for BER simulation.

- Then run the `RunBerSim()` function. This will run the transmitter, channel and receiver explained on section 6.2 `N` times (`N` is defined by the user). The range of `EbN0` values to be tested should also be selected. After the simulation chain is ran `N` times, the BER is calculated and depicted in the graph format.

A.4 USRP Co-simulation

This section presents a small user guide, explaining how to run the USRP co-simulation. The setup required is presented in Figure A.6. Two USRPs and two computers are necessary.

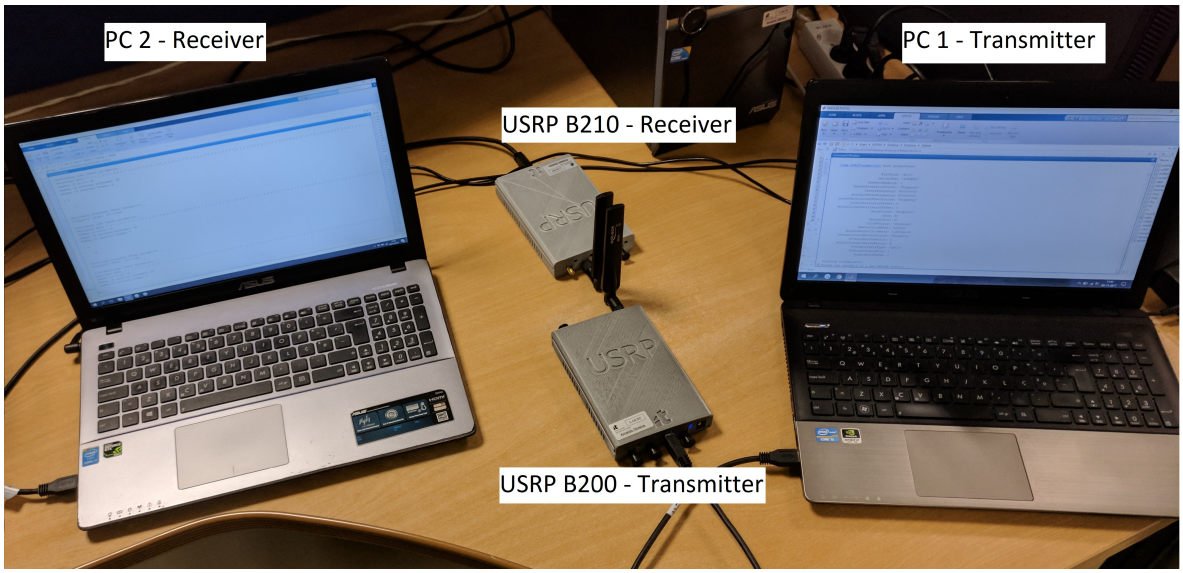


Figure A.6: Co-simulation laboratorial setup.

The several steps necessary to run the co-simulation are explained below:

- First, connect the USRP B200 on the computer chosen to transmit. Then, open the MATLAB program `sdruTx()`. On the top of the program, there is a list of parameters whose value can be changed. The parameters are the same as in Figure A.1. Finally, run the `sdruTx()` program. This program is always the first to be run.

- Then, on the receiver side, connect the USRP B210 on the computer chosen to receive. Then, open the MATLAB program `sdruRx()`. On top of the program, there is a list of parameters, whose values can be changed. The parameters are the same as in Figure A.1. Chosen parameters need to be equal on the transmitter and receiver sides, so the transport block can be decoded properly. Finally, run the `sdruRx()` program. This program is always the second one to be run.

- Finally, outputs similar to the ones from Figures A.2 and A.3 appear on each computer, for NPUSCH format 1 and 2, respectively. On the command window of the computer used to transmit, appears the transmitted data. On the command window of the computer chosen to receive, appears the remaining information. When running this co-simulation, transmitted and received constellations/eye diagrams are shown, each one in its respective computer.

Appendix B

Reference Sequence Test

This appendix shows what happens to a known sequence in each coding and modulation step. It can be used as a reference for future testing. First, a sequence for NPUSCH format 1 is tested on section B.1. Then, a sequence for NPUSCH format 2 is evaluated on section B.2.

B.1 NPUSCH Format 1

The parameters used to run this simulation were: `deltaf` = 15000, `Isc` = 18, `Irep` = 0, `Iru` = 0, `Imcs` = 0, `RVdci` = 0, `RNTI` = 1, `group_hopping` = 1, `mode` =1 and `EbN0` = 8;

Bits to be send are usually randomly selected. Here, a know sequence was imposed so it is possible to verify each step response. The selected sequence is represented below:

```
data_tx = [1 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1];
```

Afterwards, the CRC is calculated and appended (CRC addition step):

```
datacrc_tx = [1 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1
    0 1 0 1 0];
```

Then, the sequence is turbo coded. The three outputs (systematic and parity bits) are:

```
d0_tx = [1 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1
    0 1 0 0 0 1 0];

d1_tx = [1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 1 1 0 1 0 0 1 1
    1 1 0 0 0 0 1];

d2_tx = [1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 1 0
    1 0 1 0 0 1 1];
```

The next step is rate matching. The values obtained on the output of this step are depicted below:

```
codeword_tx = [1 0 1 0 1 1 0 1 0 1 0 0 0 1 1 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 0
    0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1
    0 1 1 0 0 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 0
    1 0 1 0 0 0];
```

Afterwards, the modulation processing starts. The first step consists in scrambling, with the result presented below:

```
btilde_tx = [1 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 0 1 1 1 1 0 0 1 0 0 1 1 0 1 1 0
    1 1 0 1 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1
    0 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0
    1 1 0 1 0];
```

Next, the bits are QPSK modulated as it's possible to verify:

```
symbol_tx =
[-0.7071 + 0.7071i   0.7071 + 0.7071i  -0.7071 - 0.7071i   0.7071 - 0.7071i
 -0.7071 - 0.7071i   0.7071 - 0.7071i  -0.7071 + 0.7071i   0.7071 - 0.7071i
 -0.7071 + 0.7071i  -0.7071 + 0.7071i  -0.7071 - 0.7071i  -0.7071 - 0.7071i
  0.7071 + 0.7071i  -0.7071 + 0.7071i   0.7071 - 0.7071i  -0.7071 + 0.7071i
 -0.7071 - 0.7071i   0.7071 - 0.7071i  -0.7071 + 0.7071i  -0.7071 + 0.7071i
 -0.7071 - 0.7071i  -0.7071 + 0.7071i   0.7071 - 0.7071i   0.7071 + 0.7071i
  0.7071 - 0.7071i  -0.7071 + 0.7071i   0.7071 + 0.7071i  -0.7071 - 0.7071i
  0.7071 + 0.7071i  -0.7071 + 0.7071i   0.7071 - 0.7071i  -0.7071 + 0.7071i
  0.7071 + 0.7071i  -0.7071 - 0.7071i   0.7071 - 0.7071i   0.7071 - 0.7071i
 -0.7071 + 0.7071i  -0.7071 + 0.7071i  -0.7071 + 0.7071i   0.7071 - 0.7071i
  0.7071 - 0.7071i   0.7071 - 0.7071i   0.7071 - 0.7071i  -0.7071 - 0.7071i
 -0.7071 - 0.7071i   0.7071 + 0.7071i   0.7071 - 0.7071i   0.7071 + 0.7071i
  0.7071 - 0.7071i  -0.7071 - 0.7071i   0.7071 + 0.7071i   0.7071 - 0.7071i
 -0.7071 + 0.7071i   0.7071 + 0.7071i  -0.7071 - 0.7071i   0.7071 + 0.7071i
 -0.7071 - 0.7071i   0.7071 - 0.7071i  -0.7071 + 0.7071i  -0.7071 + 0.7071i];
```

The last step consists on the SC-FDMA modulation. As the result is very long (length = 1920), only the first and last 16 samples are shown:

```
signal_tx =
[-0.1793 - 0.1441i  -0.1769 - 0.1530i  -0.1673 - 0.1531i  -0.1516 - 0.1452i
 -0.1312 - 0.1304i  -0.1077 - 0.1103i  -0.0829 - 0.0870i  -0.0584 - 0.0624i
 -0.0356 - 0.0387i  -0.0159 - 0.0175i  -0.1532 + 0.1532i  -0.0990 + 0.1683i
 -0.0329 + 0.1766i   0.0383 + 0.1762i   0.1065 + 0.1661i   0.1643 + 0.1455i     ...

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i];
```

The transmitter chain is now complete.

Then, the signal goes through the simulated AWGN channel:

```
signal_rx =
[-0.1580 - 0.1404i   -0.1264 - 0.0570i   -0.1799 - 0.1210i   -0.1239 - 0.1266i
 -0.0933 - 0.1376i   -0.1011 - 0.0980i   -0.0965 + 0.0075i   -0.0855 - 0.0855i
 -0.0938 - 0.0498i   -0.0296 - 0.0402i   -0.1452 + 0.1326i   -0.1084 + 0.1597i
 -0.0220 + 0.2075i    0.0406 + 0.1806i    0.1400 + 0.1280i    0.1327 + 0.2281i  ...

 -0.1110 + 0.0524i    -0.0959 - 0.0013i   -0.0534 + 0.0951i   0.0040 - 0.0060i
  0.0101 - 0.0005i    -0.0366 - 0.0112i    0.0656 + 0.0642i   0.0567 + 0.0527i
 -0.0089 + 0.0093i     0.1663 - 0.0250i   -0.0183 - 0.0188i   0.0492 - 0.0182i
 -0.0032 + 0.0341i    -0.0862 + 0.0507i    0.0940 + 0.0823i  -0.0532 + 0.0519i];
```

Now, the receiver chain starts. The signal is SC-FDMA demodulated:

```
symbol_rx =
[0.0659 + 0.4481i   0.1678 + 0.8830i  -0.2799 - 0.8085i   0.1903 - 0.7476i
-0.2531 - 0.7922i   0.0476 - 0.9272i  -0.1832 + 0.7531i   0.3644 - 0.5849i
-0.2274 + 0.5229i  -1.0858 + 0.9753i  -0.2888 - 1.0861i  -0.8509 - 0.5613i
 0.2875 + 0.6990i  -0.6419 + 0.5106i   0.5744 - 0.4670i  -0.7311 + 0.2196i
-0.8179 - 0.1479i   0.9289 - 1.3753i  -0.8218 + 1.4590i  -0.6888 + 0.3791i
-0.8577 - 0.4904i  -0.6301 + 0.4254i   0.6509 - 0.5591i   0.7783 + 0.3342i
-0.2517 - 0.4262i  -0.1865 + 0.6083i   0.1841 + 0.8750i  -0.0710 - 0.7323i
 0.1526 + 0.5764i   0.0327 + 0.8499i   0.1633 - 0.9297i  -0.1403 + 0.9191i
 0.1192 + 0.7870i  -0.3150 - 0.7883i   0.1634 - 0.6940i   1.2110 - 0.6684i
-0.2078 + 0.5837i  -0.9622 + 0.6124i  -0.5604 + 0.8811i   0.5798 - 0.7977i
 1.0281 - 0.5984i   0.5312 - 0.7976i   0.7328 - 0.5147i  -0.9000 - 0.8078i
-0.9002 - 0.5555i   0.8186 + 0.5800i   0.8860 - 0.5309i   0.9111 + 0.4759i
 0.9561 - 0.8479i  -1.1150 - 0.6987i   1.1357 + 0.7949i   0.4892 - 1.0180i
-0.6472 + 0.9019i   0.6985 + 0.6420i  -0.7645 - 0.7243i   0.6498 + 0.6620i
-0.4935 - 0.7417i   0.5082 - 0.6323i  -0.3292 + 0.6109i  -1.3733 + 1.0386i];
```

Afterwards, it is QPSK demodulated using soft decision:

```
btilde_rx =
[0.0659   0.4481   0.1678   0.8830   -0.2799   -0.8085    0.1903    -0.7476
-0.2531  -0.7922   0.0476  -0.9272   -0.1832    0.7531    0.3644    -0.5849
-0.2274   0.5229  -1.0858   0.9753   -0.2888   -1.0861   -0.8509    -0.5613
 0.2875   0.6990  -0.6419   0.5106    0.5744   -0.4670   -0.7311     0.2196
-0.8179  -0.1479   0.9289  -1.3753   -0.8218    1.4590   -0.6888     0.3791
-0.8577  -0.4904  -0.6301   0.4254    0.6509   -0.5591    0.7783     0.3342
-0.2517  -0.4262  -0.1865   0.6083    0.1841    0.8750   -0.0710    -0.7323
 0.1526   0.5764   0.0327   0.8499    0.1633   -0.9297   -0.1403     0.9191
 0.1192   0.7870  -0.3150  -0.7883    0.1634   -0.6940    1.2110    -0.6684
-0.2078   0.5837  -0.9622   0.6124   -0.5604    0.8811    0.5798    -0.7977
 1.0281  -0.5984   0.5312  -0.7976    0.7328   -0.5147   -0.9000    -0.8078
-0.9002  -0.5555   0.8186   0.5800    0.8860   -0.5309    0.9111     0.4759
 0.9561  -0.8479  -1.1150  -0.6987    1.1357    0.7949    0.4892    -1.0180
-0.6472   0.9019   0.6985   0.6420   -0.7645   -0.7243    0.6498     0.6620
-0.4935  -0.7417   0.5082  -0.6323   -0.3292    0.6109   -1.3733     1.0386];
```

Then, it is descrambled:

```
codeword_rx =
[0.0659    0.4481    0.1678    0.8830   -0.2799   -0.8085    0.1903   -0.7476
-0.2531   -0.7922    0.0476   -0.9272   -0.1832    0.7531    0.3644   -0.5849
 0.2274    0.5229   -1.0858    0.9753   -0.2888   -1.0861   -0.8509   -0.5613
 0.2875    0.6990   -0.6419    0.5106    0.5744   -0.4670   -0.7311    0.2196
-0.8179   -0.1479    0.9289   -1.3753   -0.8218    1.4590   -0.6888    0.3791
-0.8577   -0.4904   -0.6301    0.4254    0.6509   -0.5591    0.7783    0.3342
-0.2517   -0.4262   -0.1865    0.6083    0.1841    0.8750   -0.0710   -0.7323
 0.1526    0.5764    0.0327    0.8499    0.1633   -0.9297   -0.1403    0.9191
-0.1192    0.7870   -0.3150   -0.7883    0.1634   -0.6940    1.2110   -0.6684
-0.2078    0.5837   -0.9622    0.6124   -0.5604    0.8811    0.5798   -0.7977
 1.0281   -0.5984    0.5312   -0.7976    0.7328   -0.5147   -0.9000   -0.8078
 0.9002   -0.5555    0.8186    0.5800    0.8860   -0.5309    0.9111    0.4759
-0.9561   -0.8479   -1.1150   -0.6987    1.1357    0.7949    0.4892   -1.0180
 0.6472    0.9019    0.6985    0.6420   -0.7645   -0.7243    0.6498    0.6620
 0.4935   -0.7417    0.5082   -0.6323   -0.3292    0.6109   -1.3733    1.0386};
```

The next step consists on rate dematching:

```
d0_rx = [-0.2799 0.6419 0.5849 -1.4590 0.4481 0.5613 0.1832 0.9289 -0.7476 -0.4670
    -1.0858 0.8577 0 0.2888 -0.7922 -0.2196 0.8830 -0.6990  -0.3644  -0.8218 0.0659
    0.8509 0.9272 0.1479 0.1903 -0.5744 0.5229 0.3791 0 1.0861 0.0476 0.8179
    -0.8085 0.5106 -0.2274 0.6888 -0.1678 0.2875 -0.7531 1.3753 0.2531 0.7311
    -0.9753 0.4904];

d1_rx = [-0.0710 -1.1150  -0.5604  0 0.2517  -0.8860 1.2110 0.6323 0.1633 0.6472
    0.5312 0 0.6301 0.9000 -0.1192 -0.7645 0.1841 0.9561 0.9622 1.0386 -0.7783
    0.8186 0.1634 0.7417 -0.0327 -0.4892 -1.0281 0 0.6509 -0.9002 -0.3150 0.6620
    -0.1526 1.1357 0.5798 0 -0.1865 -0.9111  -0.2078 0.6109 0.1403 0.6985 0.7328 0];

d2_rx = [-0.6498 -0.7323  -0.6987 0.8811 0   -0.4262  -0.5309 -0.6684 -0.3292
    -0.9297 0.9019 -0.7976  0   -0.4254 0.8078 0.7870 0.7243 -0.8750 0.8479 -0.6124
    0 -0.3342  -0.5800 0.6940 -0.5082 0.8499 1.0180 -0.5984 0 -0.5591  -0.5555
    0.7883 -0.4935 0.5764 -0.7949 -0.7977 0   -0.6083 0.4759 -0.5837 1.3733 0.9191
    -0.6420 -0.5147];
```

Next, turbo decoding is performed:

```
datacrc_rx = [1 0 0  1  0  0  0  0  1  1  1  0  1  0  1  1  0  1  1  1  1  0  0  0  0  1
    0  0  1  0  0  0  1  0  1  0  1  0  1  0];
```

Finally, it is CRC checked and since no errors were detected, the appended bits were removed and data was recovered:

```
errors = 0;

data_rx = [1 0 0  1  0  0  0  0  1  1  1  0  1  0  1  1];
```

To conclude, the transmitted and received data is the same. The receiver chain is complete.

B.2 NPUSCH Format 2

The parameters used to run this simulation were: `deltaf` = 15000, `NARrep` = 0, `ACKNACK` = 5 and `RNTI` = 1.

The HARQ-ACK value is usually randomly selected. In this example, it was imposed a positive HARQ-ACK, so it is possible to verify each step response. The selected HARQ-ACK is represented below:

```
HARQ-ACK = 1;
```

After, the channel is coded into a codeword:

```
codeword_tx = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
```

Afterwards, the modulation processing starts. The first step consists in scrambling, with the result presented below:

```
btilde_tx = [1 1 0 1 1 1 1  1  0  1  1  0  0  0  0  0];
```

Next, the bits are BPSK modulated as it's possible to verify:

```
symbol_tx =
[-0.7071 - 0.7071i  -0.7071 - 0.7071i   0.7071 + 0.7071i  -0.7071 - 0.7071i
 -0.7071 - 0.7071i  -0.7071 - 0.7071i  -0.7071 - 0.7071i  -0.7071 - 0.7071i
  0.7071 + 0.7071i  -0.7071 - 0.7071i  -0.7071 - 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i];
```

The last step consists on the SC-FDMA modulation. As the result is very long (length = 1920), only the first and last 16 samples are shown:

```
signal_tx[
-0.1261 - 0.0646i  -0.1216 - 0.0596i  -0.1156 - 0.0479i  -0.1083 - 0.0297i
-0.1001 - 0.0061i  -0.0915 + 0.0216i  -0.0831 + 0.0515i  -0.0756 + 0.0816i
-0.0696 + 0.1099i  -0.0655 + 0.1345i  -0.0884 - 0.0884i  -0.0920 - 0.0971i
-0.0924 - 0.1040i  -0.0896 - 0.1091i  -0.0841 - 0.1125i  -0.0765 - 0.1144i ...

 0.0638 - 0.1539i   0.0768 - 0.1539i   0.0898 - 0.1536i   0.1019 - 0.1524i
 0.1121 - 0.1498i   0.1194 - 0.1453i   0.1230 - 0.1385i   0.1226 - 0.1294i
 0.1178 - 0.1178i   0.1088 - 0.1041i   0.0962 - 0.0886i   0.0807 - 0.0720i
 0.0634 - 0.0551i   0.0455 - 0.0387i   0.0283 - 0.0236i   0.0128 - 0.0105i];
```

The transmitter chain is complete. The signal goes through the simulated AWGN channel:

```
signal_rx
[-0.0907 - 0.0958i   -0.0782 - 0.0578i   -0.1334 - 0.1003i   -0.0557 - 0.0429i
 -0.1086 + 0.0188i   -0.0849 - 0.0268i   -0.1444 + 0.0027i   -0.0065 - 0.0241i
 -0.0273 + 0.1026i   -0.1182 + 0.1953i   -0.1542 - 0.1746i   -0.0253 - 0.0570i
 -0.0861 - 0.0597i   -0.1535 - 0.1102i   -0.0306 - 0.1179i   -0.0694 - 0.1083i...

  0.0352 - 0.1790i    0.0408 - 0.1258i    0.0597 - 0.1446i    0.0783 - 0.1456i
  0.1125 - 0.2210i    0.1001 - 0.1410i    0.2579 - 0.1314i    0.0849 - 0.1462i
  0.1353 - 0.0609i    0.1467 - 0.2066i    0.0407 - 0.1138i    0.0105 - 0.0371i
 -0.0758 + 0.0216i    0.0239 - 0.0283i    0.0541 + 0.0127i    0.0912 + 0.0006i];
```

Now, the receiver chain starts. The signal is SC-FDMA demodulated:

```
symbol_rx =
[-0.7098 - 0.7115i   -0.7067 - 0.7075i   0.7071 + 0.7054i  -0.7064 - 0.7064i
 -0.7095 - 0.7098i   -0.7059 - 0.7044i  -0.7083 - 0.7074i  -0.7046 - 0.7047i
  0.7054 + 0.7049i   -0.7046 - 0.7053i  -0.7077 - 0.7085i   0.7078 + 0.7084i
  0.7042 + 0.7057i    0.7087 + 0.7096i   0.7063 + 0.7070i   0.7095 + 0.7099i];
```

Afterwards, it is QPSK demodulated using soft decision:

```
btilde_rx =[-0.7992 -0.7957   0.7962  -0.7954  -0.7989  -0.7948  -0.7976  -0.7934
    0.7943  -0.7934  -0.7969   0.7970   0.7929   0.7980   0.7953   0.7989];
```

Then, it is descrambled in hard decision format:

```
codeword_rx = [1 1 1  1  1  1  1  1  1  1  1  1  1  1  1  1];
```

Finally, an HARQ-ACK is recovered:

```
HARQ-ACK = 1;
```

To conclude, the transmitted and received HARQ-ACK value is the same. The receiver chain is complete.